

The eXSchedule tool is our approach to provide scheduling of appointments into your application. The ExSchedule library lets the user changes its visual appearance using skins, each one providing an additional visual experience that enhances viewing pleasure.

Features include:

- Skinnable Interface support (ability to apply a skin to any background part)
- Undo/Redo support
- XML data support
- **<u>Unlimited</u>** options to show any HTML text, images, colors, EBNs, patterns, frames anywhere on the event's background.
- Easy way to define the control's visual appearance in design mode, using XP-Theme elements or EBN objects
- Ability to enlarge or **zooming** the scheduler, from one year to minutes, using the middle mouse button
- Ability to smoothly navigate from one date to another by dragging the scheduler
- Any major UI part of the scheduler can be **drag and drop** to any other side
- Ability to customize the mouse+keys combinations for almost all UI operations such as moving, resizing the events, zooming, and so on
- **Multiple selection** support, or ability to remove, move or resize all selected appointments/events at once
- Multiple HTML lines Tooltip support
- Display Multiple Schedules Side-by-Side, or ability to group or filter the events
- Ability to define the one or more **non-working** parts of any day, week, month, ... with a different pattern, colors, and so on, based on custom expressions
- **Repetitive** events support, or scheduling something that occurs at the same time multiple days
- Ability to mark one or more zones with a different color, HTML text, pattern and so on
- Define one or **more timers**, with the ability to change the UI attributes of meeting events
- Integrated Time Scales support, or ability to display one or more time scales, using different time zones
- All-Day Header support, or an all-day event that stretches over multiple days can be displayed as a contiguously bar across those days
- Ability to specify **disabled dates** (so the user can't select them), based on custom expressions, or you can specify the range of date to display data from
- Any Event can display one or more multiple-lines HTML captions, one or more pictures, icons, aligned to any part of the event
- The event's labels or tooltips may use custom expressions that shows the data such as

duration automatically once the event is moved or resized.

- The Event may display the status or the body part, using different colors, EBN, patterns, and so on
- You can specify whether an event is selectable, movable, resizable to one or both sides, what's are its margins or limits, and so on
- Single / Multi-Day Appointment/Event support



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

How to get support?

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

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

- Check out the samples they are here to provide some quick info on how things should be done
- Check out the how-to questions using the <u>eXHelper</u> 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 <u>here</u>.
- 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 Column object uses the AlignmentEnum enumeration to align a column.

Name	Valu	e Description
LeftAlignment	0	The source is left aligned.
CenterAlignment	1	The source is centered.
RightAlignment	2	The source is right aligned.

constants AllDayEventScrollEnum

The <u>AllowAllDayEventScroll</u> property gets or sets a value that specifies whether the all-day event header supports scrolling. Use the <u>ShowAllDayHeader</u> property to show the schedule's All-Day header so all All-Day evens are shown on this header. The <u>AllowCreateAllDayEvent</u> property has effect only when the schedule displays no time scale. The AllDayEventScrollEnum type supports the following flags:

Name	Value	Description
exAllDayEventNoScroll	0	The all-day event header supports no scroll.
exAllDayEventNoMin	0	The all-day event header displays events as many as possible.
exAllDayEventMin4	4	The all-day event header displays at least 4 events. This flag can be combined with exAllDayEventScroll, exAllDayEventWheelScroll or any Max flag
exAllDayEventMin8	8	The all-day event header displays at least 8 events. This flag can be combined with exAllDayEventScroll, exAllDayEventWheelScroll or any Max flag
exAllDayEventMin12	12	The all-day event header displays at least 12 events. This flag can be combined with exAllDayEventScroll, exAllDayEventWheelScroll or any Max flag
exAllDayEventNoMax	16	The all-day event header displays events as many as possible.
exAllDayEventMax4	64	The all-day event header displays at most 4 events. This flag can be combined with exAllDayEventScroll, exAllDayEventWheelScroll or any Min flag
exAllDayEventMax8	128	The all-day event header displays at most 8 events. This flag can be combined with exAllDayEventScroll, exAllDayEventWheelScroll or any Min flag
exAllDayEventMax12	192	The all-day event header displays at most 12 events. This flag can be combined with exAllDayEventScroll, exAllDayEventWheelScroll or any Min flag
exAllDayEventScroll	256	The all-day event header supports scrolling.

The user can scroll the all-day event header by rotating the mouse wheel, while the cursor hovers the header. Remove this flag to disable scrolling using the mouse wheel.

constants AllowKeysEnum

The AllowKeysEnum type specifies the keys to be combined in order to start an UI operation. For instance, the <u>AllowCreateEvent</u> property of AllowKeysEnum type indicates the keys combination to let user creates a new event at runtime. By default, this property is set on exLeftClick, which means the user is able to create a new event by single left click. If this property is set on exRightClick + exCRTLKey the user should press the CTRL key while right clicking the control to start creating a new event. If the exDblClick flag is included, the user requires to do a double click instead single click to perform the operation. The exDisallow flag indicates that the operation is not allowed.

Name	Valu	e Description
exDisallow	0	The operation is not allowed.
exLeftClick	1	The operation starts once the user clicks the left mouse button.
exRightClick	2	The operation starts if the user clicks the right mouse button.
exMiddleClick	3	The operation starts if the user clicks the middle mouse button.
exSHIFTKey	8	The operation may start only if the user presses the SHIFT key.
exCTRLKey	16	The operation may start only if the user presses the CTRL key.
exALTKey	32	The operation may start only if the user presses the ALT key.
exDblClick	64	The operation starts only if the user double clicks, instead single click.

The AllowKeysEnum type supports the following values:

constants AppearanceEnum

The AppearanceEnum type indicates the type of borders the control can show. The <u>Appearance</u> property indicates the border the scheduler displays. The Appearance property supports EBN objects, so actually, you can define your own type of border. The Appearance property supports the following values:

Name	Value	e 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 BackgroundExtPropertyEnum

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

Name

exToStringExt

Value Description

Specifies the part's ToString representation. The <u>BodyBackgroundExt</u> property specifies the EBN String format to be displayed on the object's background. *The Exontrol's <u>eXButton</u> WYSWYG* Builder helps you to generate or view the EBN String Format, in the **To String** field.

Sample:

0

"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).

		Indicates the background color / EBN color to be shown on the part of the object. <i>Sample: 255 indicates red, RGB(0,255,0) green, or 0x1000000.</i>
exBackColorExt	1	(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 the (string or numeric) them.

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.

- 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.

exClientExt

2

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**
- <i> ... </i> displays the text in *italics*
- <u> ... </u> <u>underlines</u> 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

exAnchorExt

3

FormatAnchor property customizes the visual effect for anchor elements.

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

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

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

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

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

4

exTextExt

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.

- key[:width] inserts a custom size picture into the text being previously loaded using the HTMLPicture property. The Key parameter indicates the key of the picture being displayed. The Width parameter indicates a custom size, if you require to stretch the picture, else the original size of the picture is used.
- & glyph characters as & amp; (&), < (<), > (>), &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 FFFFF;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 "<font;31><sha 404040;5;0>

<fgcolor=FFFFFF>outline antialiasing</fgcolor></**sha**>" gets:

outline anti-aliasing

(String expression)

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 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.

exTextExtAlignment

exTextExtWordWrap

6

5

- 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**),
- 8, (hexa 0x008, **Vertical**),
- 9, (hexa 0x009, Horizontal),
- 10, (hexa 0x00A, **Cross**),
- 11, (hexa 0x00B, **Brick**),

exPatternExt

7

		 12, (hexa 0x00C, Yard), 256, (hexa 0x100, Frame), 256, (hexa 0x100, Frame), <i>c</i> The exFrameColorExt specifies the color to show the frame. The Frame flag can be combined with any other flags. 768, (hexa 0x300, FrameThick), <i>c</i> The exFrameColorExt specifies the color to show the frame. The Frame flag can be combined with any other flags.
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) <i>(Color expression)</i>
exFrameColorExt	9	Indicates the color to show the border-frame on the object. This property set the Frame flag for exPatternExt property. <i>(Color expression)</i>
exFrameThickExt	10	Specifies that a thick-frame is shown around the object. This property set the FrameThick flag for exPatternExt property. <i>(Boolean expression)</i>
exUserDataExt	11	Specifies an extra-data associated with the object. (<i>Variant expression</i>)

constants BackgroundPartEnum

The BackgroundPartEnum type indicates the UI parts of the control, whose background or foreground colors can be changed. The background can be a solid color or an EBN object. The <u>Background</u> property of the control specifies the background/foreground color of a specified UI part of the control. The /NET or /WPF version of the control provides the get_Background/get_Background32 properties to get a color for a specified UI part, and set_Background/set_Background32 for changing the part's background or foreground colors. The eXSchedule tool may display two panels, the calendar and the schedule panel.

The *options that starts with exCalendar* refer to a part of the Calendar panel as shown bellow:



while if the option starts with exSchedule it refers an UI part of the schedule panel as:

	Thursday, J	une 21, 2012	Friday, Ju	une 22, 2012	
08:00 AM :30 09:00 AM	9:00 AM - 10:45 A	M	8:30 AM - 9:45 AM		
10:00 AM :30			10:00 AM - 11:00 AM	9:30 AM - 12:45	
11:00 AM :30 12:00 PM	10:45 AM - 12:30 PM	11:15 AM - 1:45 PM	11:00 AM - 1:45 PM		t.
:30 01:00 PM :30 02:00 PM	1:30 PM - 3:15 PM			1:15 PM - 2:45 PM	
:30 03:00 PM :30					

The BackgroundPartEnum type supports the following values.

	Name	Valu	le Description
exCalendarArrowUp 0 Specifies the visual appearance for the up arrow the calendar's header.	exCalendarArrowUp	0	Specifies the visual appearance for the up arrow in the calendar's header.

exCalendarArrowDown	1	Specifies the visual appearance for the down arrow in the calendar's header.
exCalendarArrowLeft	2	Specifies the visual appearance for the left arrow in the calendar's header. You can use the <u>MinDate</u> property to limit the dates that the calendar can show. When the calendar has no other dates to show, the left or right arrows are not shown. The <u>MaxDate</u> property indicates the upper margin that the calendar can show.
exCalendarArrowRight	3	Specifies the visual appearance for the right arrow in the calendar's header. You can use the <u>MinDate</u> property to limit the dates that the calendar can show. When the calendar has no other dates to show, the left or right arrows are not shown. The <u>MaxDate</u> property indicates the upper margin that the calendar can show.
exCalendarBackColor	4	Specifies the calendar's background color. The Background(exCalendarBackColor) property changes the calendar's panel backcolor if not-zero, else the <u>BackColor</u> property specifies the calendar's background color.
exCalendarForeColor	5	Specifies the calendar's foreground color. The Background(exCalendarForeColor) property changes the calendar's panel foreground color. If this option is not set, the control's <u>ForeColor</u> property indicates the calendar's foreground color.
exCalendarDaysHeader	6	Specifies the visual appearance for the days header.
exCalendarWeeksHeader	7	Specifies the visual appearance for the weeks header.
exCalendarHeader	8	Specifies the visual appearance for the months header in the calendar panel.
exCalendarTodayUp	9	Specifies the visual appearance for the today button in the calendar panel, when it is up. Use the <u>ShowTodayButton</u> property to hide the Today button.
exCalendarTodayDown	10	Specifies the visual appearance for the today button in the calendar panel, when it is down. Use the <u>ShowTodayButton</u> property to hide the Today button.

exCalendarScrollThumb	11	Specifies the visual appearance for the scrolling thumb in the calendar panel. The <u>ShowYearScroll</u> property indicates whether the calendar panel displays an horizontal scroll bar to allow the user to change the calendar's year.
exCalendarScrollRange	12	Specifies the visual appearance for the scrolling range in the calendar panel. The <u>ShowYearScroll</u> property indicates whether the calendar panel displays an horizontal scroll bar to allow the user to change the calendar's year.
exCalendarSplitBar	13	Specifies the visual appearance for the separator bar in the calendar panel.
exCalendarMarkToday	14	Returns or sets a value that indicates the visual appearance for Today date, in the calendar panel. The exScheduleMarkTodayBackColor option changes the visual appearance for Today date, in the schedule panel.
exCalendarMonthSelect	15	Specifies the visual appearance for the selected month in the months drop down window.
exCalendarMonthSelectFore	C olo r	Specifies the foreground color for the selected month in the months drop down window.
exCalendarGridLineColor	17	Specifies the color to show the calendar's grid lines. The <u>ShowGridLines</u> property indicates the type of grid lines to be shown in the calendar.
exSplitBar	18	Specifies the visual appearance for control's vertical split bar.
exCalendarSelBackColor	19	Specifies the calendar's background color for selected dates. The <u>AllowSelectDate</u> property specifies the keys combination that user can use to select dates in the calendar panel. The <u>AllowSelectDateRect</u> specifies the keys combination so the user can do a rectangular selection in the calendar panel.
exCalendarSelForeColor	20	Specifies the calendar's foreground color for selected dates. The <u>AllowSelectDate</u> property specifies the keys combination that user can use to select dates in the calendar panel. The <u>AllowSelectDateRect</u> specifies the keys combination so the user can do a rectangular

		selection in the calendar panel.
exCalendarHeaderForeColor 2	21	Specifies the foreground color to show the months in the header.
exCalendarBorderLineColor 2	26	exCalendarBorderLineColor. Specifies the color to show the calendar's border lines.
exCalendarCommentDate 2	27	exCalendarCommentDate. Specifies the visual appearance to show the dates in the calendar with a tooltip assigned.
exCalendarDaysHeaderForeCa	2186 r	Specifies the foreground color for the days header.
exCalendarWeeksHeaderFore	2 20 lor	Specifies the foreground color for the weeks header.
exCalendarMarkTodayForeCol	6 6	Specifies the foreground color for the today date, in the calendar panel. The exScheduleMarkTodayForeColor option changes the visual appearance for Today date, in the schedule panel.
exCalendarTodayForeColor	31	Specifies the foreground color for the Today button, in the calendar panel. Use the <u>ShowTodayButton</u> property to hide the Today button.
exCalendarFocusDate	32	Specifies visual appearance for the focused date, in the calendar panel. The <u>AllowFocusDate</u> property on exDisallow disables focusing a date that's not being selected.
exCalendarFocusDateForeCol	8 6	Specifies foreground color for the focused date, in the calendar panel. The <u>AllowFocusDate</u> property on exDisallow disables focusing a date that's not being selected.
exScheduleBorderDateColor	34	Specifies color to display the border for the dates.
exScheduleBorderMonthColor	35	Specifies color to display the border for the months.
exScheduleBorderSelColor	36	Specifies color to display the border for selected dates.
exScheduleBorderSelColorUnR	307cus	Specifies color to display the border for selected dates, when the control has not focus.
exScheduleBorderTimeScaleCa	8 &r	Specifies the color to display the border for time scales.
exScheduleTimeScaleBackCol	9 9	Specifies the color to display the time scale's default background color.

exScheduleTimeScaleForeCold40	Specifies the foreground color to display the time scale.
exScheduleDayHeaderBackCo4dr	Specifies the visual appearance of the header's day in the schedule view.
exScheduleDayHeaderForeCo#2	Specifies the foreground color of the header's day in the schedule view.
exScheduleDayGroupBackCol 4 6	Specifies the visual appearance of the group in the header's day of the schedule view.
exScheduleDayGroupForeColdr4	Specifies the foreground color of the group in the header's day of the schedule view.
exScheduleDayTimeBackColor45	Specifies the visual appearance of the time scale in the day of the schedule view.
exScheduleDayTimeForeColor46	Specifies the foreground color of the time scale in the day of the schedule view.
exScheduleTimeScaleRulerBack	oSpecifies the visual appearance of the time ruler.
exScheduleTimeScaleRulerFor	orSpecifies the foreground color of the time ruler.
exScheduleTimeScaleMajorRu#9Cc	Specifies the color to show the line of the major olor time ruler.
	Specifies the style of the line of the major time ruler.
exScheduleTimeScaleMajorRu f@ St	The Background(exScheduleTimeScaleMajorRulerStyle) property indicates a <u>LinesStyleEnum</u> expression that determines style of lines to be shown on major rulers
exScheduleTimeScaleMajorRuterSt	The Background(exScheduleTimeScaleMajorRulerStyle) property indicates a <u>LinesStyleEnum</u> expression that determines style of lines to be shown on major rulers Specifies the color to show the line of the minor time ruler.
exScheduleTimeScaleMajorRuterSt exScheduleTimeScaleMinorRuterCo exScheduleTimeScaleMinorRuterSt	The Background(exScheduleTimeScaleMajorRulerStyle) property indicates a <u>LinesStyleEnum</u> expression that determines style of lines to be shown on major rulers Specifies the color to show the line of the minor time ruler.
exScheduleTimeScaleMajorRuterState exScheduleTimeScaleMinorRuterCot exScheduleTimeScaleMinorRuterState	The Background(exScheduleTimeScaleMajorRulerStyle) Property indicates a LinesStyleEnum expression that determines style of lines to be shown on major rulers Specifies the color to show the line of the minor time ruler. yleSpecifies the style of the line of the minor time ruler. Specifies the color to show the line of the major time ruler in the schedule panel.
exScheduleTimeScaleMajorRuterState exScheduleTimeScaleMinorRuterCot exScheduleTimeScaleMinorRuterState exScheduleMajorTimeRulerCot	The Background(exScheduleTimeScaleMajorRulerStyle) property indicates a LinesStyleEnum expression that determines style of lines to be shown on major rulers Specifies the color to show the line of the minor time ruler. (ISpecifies the style of the line of the minor time ruler. Specifies the color to show the line of the major time ruler. Specifies the style of the line of the major time ruler. Specifies the style of the line of the major time ruler in the schedule panel.
exScheduleTimeScaleMajorRuterState exScheduleTimeScaleMinorRuterState exScheduleTimeScaleMinorRuterState exScheduleMajorTimeRulerCoto	The Background(exScheduleTimeScaleMajorRulerStyle) property indicates a LinesStyleEnum expression that determines style of lines to be shown on major rulers Specifies the color to show the line of the minor time ruler. (Specifies the style of the line of the minor time ruler. Specifies the color to show the line of the major time ruler in the schedule panel. Specifies the style of the line of the major time ruler in the schedule panel. Specifies the color to show the line of the major time ruler in the schedule panel.
exScheduleTimeScaleMajorRuteOSt exScheduleTimeScaleMinorRute2St exScheduleTimeScaleMinorRute2St exScheduleMajorTimeRulerCotoo exScheduleMajorTimeScaleSt ot exScheduleMinorTimeScaleSt otoo	The Background(exScheduleTimeScaleMajorRulerStyle) property indicates a LinesStyleEnum expression that determines style of lines to be shown on major rulers Specifies the color to show the line of the minor time ruler. (Depecifies the style of the line of the minor time ruler. Specifies the color to show the line of the major time ruler in the schedule panel. Specifies the style of the line of the major time ruler in the schedule panel. Specifies the color to show the line of the minor time ruler in the schedule panel. Specifies the style of the line of the minor time ruler in the schedule panel. Specifies the style of the line of the minor time ruler in the schedule panel.

	exScheduleGroupingButton	58	Specifies the visual appearance for the drop down grouping button.
	exGroupingBackColor	59	Specifies the background color for the drop down grouping view.
	exGroupingForeColor	60	Specifies the foreground color for the drop down grouping view.
	exGroupingSelBackColor	61	Specifies the visual appearance to display the selected items in the drop down grouping view.
	exGroupingSelForeColor	62	Specifies the foreground color to show the selected items in the drop down grouping view.
	exToolTipAppearance	64	Specifies the visual appearance of the borders of the tooltips.
	exToolTipBackColor	65	Specifies the tooltip's background color.
	exToolTipForeColor	66	Specifies the tooltip's foreground color.
	exCalendarSelBackColorUnFo	0 68 s	Specifies the background color for selected object when the control loses the focus.
exCalendarSelForeColorUnFo@s		afta)	Specifies the foreground color for selected object when the control loses the focus.
	exCheckBoxState0	70	Specifies the visual appearance for the check box in 0 state (unchecked).
	exCheckBoxState1	71	Specifies the visual appearance for the check box in 1 state (checked).
	exCheckBoxState2	72	Specifies the visual appearance for the check box in 2 state (partial, not used).
	exRadioButtonState0	73	Specifies the visual appearance for the radio button in 0 state (unchecked).
	exRadioButtonState1	74	Specifies the visual appearance for the radio button in 1 state (checked).
exScheduleCreateEventBackCoffor		Cõtor	Specifies the visual appearance of the event being created. The <u>CreateEventLabel</u> property specifies the label to be shown when the user creates a new event. The <u>CreateEventLabelAlign</u> property indicates the alignment of the label while user creates new events. The <u>Add</u> method adds programmatically a new event to the control.
			Indicates the foreground color for the event being created. The CreateEventLabel property specifies

exScheduleCreateEventForeCoor	the label to be shown when the user creates a new event. The <u>CreateEventLabelAlign</u> property indicates the alignment of the label while user creates new events. The <u>Add</u> method adds programmatically a new event to the control.
exScheduleEventContinuePrevØay	Specifies the visual appearance of the sign that's shown when the current event continues on the previously day.
exScheduleEventContinueNextD&y	Specifies the visual appearance of the sign that's shown when the current event continues on the next day.
exScheduleUpdateEventsBack720lor	Specifies the visual appearance of the event being moved or resized.
exScheduleUpdateEventsFore80lor	Indicates the foreground color for the event being moved or resized.
exScheduleMarkTodayBackCo&	Specifies the background color for the today date, in the schedule panel.
exScheduleMarkTodayForeCo	Specifies the foreground color for the today date, in the schedule panel.
exScheduleEditEventBackColo83	Specifies the background color while editing an event. The <u>Editable</u> property indicates whether the user can edit the event's label at runtime.
exScheduleEditEventForeColo84	Specifies the foreground color while editing an event. The <u>Editable</u> property indicates whether the user can edit the event's label at runtime.
exScheduleEventContinuePrev&&eek	Specifies the visual appearance of the sign that's shown when the current all-day event continues on the previously week. If this option is 0 (by default) a left-arrow icon is being displayed. You can use this option to specify the visual appearance using the EBN objects.
exScheduleEventContinueNext&ceek	Specifies the visual appearance of the sign that's shown when the current all-day event continues on the next week. If this option is 0 (by default) a right- arrow icon is being displayed. You can use this option to specify the visual appearance using the EBN objects.
exScheduleAllDayHeaderBack&olor	exScheduleAllDayHeaderBackColor. Specifies the visual appearance of the control's All-Day header.

exScheduleOLEDropPosition	97	By default, the exScheduleOLEDropPosition is 0, which means no effect. Specifies the visual appearance of the dropping position inside the schedule part of the control, when the control is implied in a OLE Drag and Drop operation. The exScheduleOLEDropPosition has effect only if different than 0, and the <u>OLEDropMode</u> property is not exOLEDropNone. For instance, set the Background(exScheduleOLEDropPosition) property on RGB(0,0,255), and a blue line is shown at the date-time position when the cursor is hover the schedule part of the control, during an OLE Drag and Drop position. The <u>OLEDragDrop</u> event notifies your application once an object is drop in the control.
exContextMenuAppearance	99	Specifies the visual appearance of the control's context menu.
exContextMenuBackColor	100	Specifies the solid background color for the control's context menu.
exContextMenuForeColor	101	Specifies the text foreground color for the control's context menu.
exContextMenuSelBackColor	102	Specifies the solid/EBN selection's background color in the control's context menu.
exContextMenuSelBorderCold	or103	Specifies the solid color to show the selection in the control's context menu.
exContextMenuSelForeColor	104	Specifies the selection's text foreground color in the control's context menu.
exScheduleDayBackColorAlte	r hā9 e	Specifies the visual appearance for alternate days.
exScheduleDayForeColorAlte	r f16te	Specifies the foreground color for alternate days.
exScheduleAllDayEventScroll	3 265 6C	Specifies the visual appearance to put on the all- day events header, when it contains scrolling bevents. The <u>AllowAllDayEventScroll</u> property gets or sets a value that specifies whether the all-day event header supports scrolling.
exVSUp	256	The up button in normal state.
exVSUpP	257	The up button when it is pressed.
exVSUpD	258	The up button when it is disabled.

exVSUpH	259	The up button when the cursor hovers it.
exVSThumb	260	The thumb part (exThumbPart) in normal state.
exVSThumbP	261	The thumb part (exThumbPart) when it is pressed.
exVSThumbD	262	The thumb part (exThumbPart) when it is disabled.
exVSThumbH	263	The thumb part (exThumbPart) when cursor hovers it.
exVSDown	264	The down button in normal state.
exVSDownP	265	The down button when it is pressed.
exVSDownD	266	The down button when it is disabled.
exVSDownH	267	The down button when the cursor hovers it.
exVSLower	268	The lower part(exLowerBackPart)in normal state.
exVSLowerP	269	The lower part(exLowerBackPart)when it is pressed.
exVSLowerD	270	The lower part(exLowerBackPart)when it is disabled.
exVSLowerH	271	The lower part (exLowerBackPart) when the cursor hovers it.
exVSUpper	272	The upper part(exUpperBackPart)in normal state.
exVSUpperP	273	The upper part(exUpperBackPart)when it is pressed.
exVSUpperD	274	The upper part(exUpperBackPart)when it is disabled.
exVSUpperH	275	The upper part (exUpperBackPart) when the cursor hovers it.
exVSBack	276	The background part (exLowerBackPart and exUpperBackPart) in normal state.
exVSBackP	277	The background part (exLowerBackPart and exUpperBackPart) when it is pressed.
exVSBackD	278	The background part (exLowerBackPart and exUpperBackPart) when it is disabled.
exVSBackH	279	The background part (exLowerBackPart and exUpperBackPart) when the cursor hovers it.

exHSLeft	384	The left button in normal state.
exHSLeftP	385	The left button when it is pressed.
exHSLeftD	386	The left button when it is disabled.
exHSLeftH	387	The left button when the cursor hovers it.
exHSThumb	388	The thumb part (exThumbPart) in normal state.
exHSThumbP	389	The thumb part (exThumbPart) when it is pressed.
exHSThumbD	390	The thumb part (exThumbPart) when it is disabled.
exHSThumbH	391	The thumb part (exThumbPart) when the cursor hovers it.
exHSRight	392	The right button in normal state.
exHSRightP	393	The right button when it is pressed.
exHSRightD	394	The right button when it is disabled.
exHSRightH	395	The right button when the cursor hovers it.
exHSLower	396	The lower part (exLowerBackPart) in normal state.
exHSLowerP	397	The lower part (exLowerBackPart) when it is pressed.
exHSLowerD	398	The lower part (exLowerBackPart) when it is disabled.
exHSLowerH	399	The lower part (exLowerBackPart) when the cursor hovers it.
exHSUpper	400	The upper part (exUpperBackPart) in normal state.
exHSUpperP	401	The upper part (exUpperBackPart) when it is pressed.
exHSUpperD	402	The upper part (exUpperBackPart) when it is disabled.
exHSUpperH	403	The upper part (exUpperBackPart) when the cursor hovers it.
exHSBack	404	The background part (exLowerBackPart and exUpperBackPart) in normal state.
exHSBackP	405	The background part (exLowerBackPart and exUpperBackPart) when it is pressed.
exHSBackD	406	The background part (exLowerBackPart and exUpperBackPart) when it is disabled.

exHSBackH	407	exUpperBackPart) when the cursor hovers it.
exSBtn	324	All button parts (L1-L5, LButton, exThumbPart, RButton, R1-R6), in normal state.
exSBtnP	325	All button parts (L1-L5, LButton, exThumbPart, RButton, R1-R6), when it is pressed.
exSBtnD	326	All button parts (L1-L5, LButton, exThumbPart, RButton, R1-R6), when it is disabled.
exSBtnH	327	All button parts (L1-L5, LButton, exThumbPart, RButton, R1-R6), when the cursor hovers it .
exScrollHoverAll	500	Enables or disables the hover-all feature. By default (Background(exScrollHoverAll) = 0), the left/top, right/bottom and thumb parts of the control' scrollbars are displayed in hover state while the cursor hovers any part of the scroll bar (hover-all feature). The hover-all feature is available on Windows 11 or greater, if only left/top, right/bottom, thumb, lower and upper-background parts of the scrollbar are visible, no custom visual-appearance is applied to any visible part. The hover-all feature is always on If Background(exScrollHoverAll) = 1 disables the hover-all feature.
exVSThumbExt	503	The thumb-extension part in normal state.
exVSThumbExtP	504	The thumb-extension part when it is pressed.
exVSThumbExtD	505	The thumb-extension part when it is disabled.
exVSThumbExtH	506	The thumb-extension when the cursor hovers it.
exHSThumbExt	507	The thumb-extension in normal state.
exHSThumbExtP	508	The thumb-extension when it is pressed.
exHSThumbExtD	509	The thumb-extension when it is disabled.
exHSThumbExtH	510	The thumb-extension when the cursor hovers it.
exScrollSizeGrip	511	Specifies the visual appearance of the control's size grip when both scrollbars are shown.

constants ChangeOperationEnum

The ChangeOperationEnum type specifies the operation performed in the scheduler control. The <u>ChangeEvent</u> event occurs when one of the following operation occurs. The ChangeOperationEnum type supports the following values:

Name	Value	Description
exAddEvent	0	Notifies once a new event is added to the scheduler. The ChangeEvent(exAddEvent) event is equivalent with the <u>AddEvent</u> event.
exRemoveEvent	1	Notifies once an event is removed from the scheduler. The ChangeEvent(exRemoveEvent) event is equivalent with the <u>RemoveEvent</u> event.
exUpdateEvent	16	Notifies once any known properties of the event (like start or ending position) is updated. The exUpdateEvent flag is combined with one of the <u>EventKnownPropertyEnum</u> value depending on what property has been changed. For instance, the exUpdateEvent + exEventEndDateTime ($16 + 2 =$ 18) indicates that the ending position of the event is changed, or the end margin of the event was resized.

constants ContentAlignmentEnum

The ContentAlignmentEnum type specifies where the object is being displayed relative to its holder. For instance, you can specify the event's label to be in the lower right margin, or in the middle. There are a lot of align properties as follows:

- <u>CreateEventLabelAlign</u>, specifies the alignment of the label to be displayed when the user creates at runtime new events.
- <u>UpdateEventsLabelAlign</u>, specifies the alignment of the label to be displayed when the user creates at runtime new events.
- Event. LabelAlign, specifies the alignment of the event's label.
- Event. ExtraLabelAlign, specifies the alignment of the event's extra label.
- Event. <u>PicturesAlign</u>, specifies the alignment of the event's pictures.
- Event. ExtraPicturesAlign, specifies the alignment of the event's extra pictures.
- TimeScale.<u>CaptionAlign</u>, specifies the alignment of the event's extra pictures.
- and so on

The ContentAlignmentEnum type supports the following values:

Name	Valu	e Description
exTopLeft	0	Content is vertically aligned at the top, and horizontally aligned on the left.
exTopCenter	1	Content is vertically aligned at the top, and horizontally aligned at the center.
exTopRight	2	Content is vertically aligned at the top, and horizontally aligned on the right.
exMiddleLeft	16	Content is vertically aligned in the middle, and horizontally aligned on the left.
exMiddleCenter	17	Content is vertically aligned in the middle, and horizontally aligned at the center.
exMiddleRight	18	Content is vertically aligned in the middle, and horizontally aligned on the right.
exBottomLeft	32	Content is vertically aligned at the bottom, and horizontally aligned on the left.
exBottomCenter	33	Content is vertically aligned at the bottom, and horizontally aligned at the center.
exBottomRight	34	Content is vertically aligned at the bottom, and horizontally aligned on the right.

constants DescriptionTypeEnum

The DescriptionTypeEnum type defines parts of the control whose default caption can be changed. The <u>Description</u> property defines the default caption to be displayed on giving UI part of the control.

Currently, the DescriptionTypeEnum type supports the following values:

Name	Value	e Description
exGroupBarAll	0	Defines the caption of '(All)' in the group bar window.

constants EditableCaptionEnum

The EditableCaptionEnum type indicates the event's property being edited when user double clicks the event. The <u>Editable</u> property indicates the property of the event to be edited when user double clicks the event. The <u>AllowEditEvent</u> property specifies the combination of keys that the user can use so the event gets inline editing. The EditableCaptionEnum type supports the following values:

Name	Valu	e Description
exNoEditable	0	The event's caption is not editable.
exEditCaption	1	The event's Caption property is editable.
exEditShortLabel	2	The event's <u>ShortLabel</u> property is editable.
exEditLongLabel	3	The event's LongLabel property is editable.
exEditExtraLabel	4	(Default). The event's <u>ExtraLabel</u> property is editable.
exEditRepetitive	5	The event's Repetitive property is editable.
exEditAcceptsReturn	16	Specifies that the ENTER key inserts new lines during edit. The exEditAcceptsReturn flag can be combined with any other value.

constants EventKnownPropertyEnum

The EventKnowPropertyEnum defines the value in the "<%=%VALUE%>" expression that can be used by label properties as follows:

- <u>DefaultEventLongLabel</u>, defines the HTML labels for events, when it fit entirely in the event's body.
- <u>DefaultEventShortLabel</u>, defines the labels for events (no HTML attribute is applied), when it does not fit the event's body
- CreateEventLabel, defines the label when creating a new event by dragging
- <u>UpdateEventsLabel</u>, defines the label of the events being moved or resized at runtime
- Event.<u>ShortLabel</u>, defines the event's short label, or the label to be shown when the LongLabel does not fit entirely the event's body. The ShortLabel displays no HTML tags
- Event.<u>LongLabel</u>, defines the event's HTML long label, when it fits the body. If the LongLabel does not fit entirely the event's body, the ShortLabel is displayed instead.
- Event.<u>ExtraLabel</u>, defines the event's extra HTML label. The event's ExtraLabel is displayed ONLY, if the LongLabel fits the event's body

For instance, the Event.ExtraLabel = "<%=%5%>, <%=%262%>", displays automatically the event's caption and the group's label where the event belongs in the bottom side of the control. If the caption or the owner group of the event is changed, the ExtraLabel is automatically updated with the new values.

The known and supported values are:

Name	Value	e Description
exEventStartDateTime	1	Indicates the starting date/time of the event. This property gets or sets the <u>Start</u> property of the event. The Start property defines the lower margin of the event, and it includes the date and the time values. The exEventStartDate specifies the DATE value only, while the exEventStartTime includes the TIME value only. For instance, the LabelProperty = "<%=weekday(%1)%>" displays the day of the week where the event starts.
		Indicates the ending date/time of the event. This property gets or sets the <u>End</u> property of the event. The End property defines the upper margin of the event, and it includes the date and the time values. The exEventEndDate specifies the DATE value only,

exEventEndDateTime	2	<pre>while the exEventEndTime includes the TIME value only. For instance, the LabelProperty = " <%=weekday(%2)%>" gets the day of the week where the event ends. (DATE expression)</pre>
exEventAllDay	3	Indicates if the current event is an all day event. This property is equivalent with the event's <u>AllDayEvent</u> property which indicates if the current event is an all-day event. This property may returns a boolean value, or 0(False) and -1(True). For instance, the LabelProperty = "<%=%3 ? `All-Day- Event: `: ``%><%=%256%>", displays automatically an "All-Day-Event: " prefix for all-day events. If the event is not an all-day event, the <%=%256%>, or exEventDisplayShortMargins, short margins of the events are displayed. (Boolean expression)
exEventGroupID	4	Indicates the identifier of the event's group. The <u>GroupID</u> property of the event indicates the identifier of the group that event belongs to. The exEventGroupLabel property indicates the <u>Caption</u> property of the Group's event. The exEventGroupTitle property indicates the <u>Title</u> property of the Group's event. For instance, the LabelProperty = " $<\%$ =%4%> $<$ br> $<\%$ =%256%>" displays on the first line, the group's identifier, and the short margins of the event on the second line. The caption of the label is automatically updated once an event is moved from a group to another. (Long expression)
exEventCaption	5	Indicates the caption of the event. The <u>Caption</u> property of the event specifies the custom caption that can be displayed on the label, without having to change the event's label. For instance, the LabelProperty = "<%=%256%> <%=%5%>" displays on the first line, the event's short margins, while on the second line displays the event's
		caption. Once you update or edit the event's Caption, the event's body automatically shows the new caption. (String expression)
-----------------	---	--
exEventUserData	6	Indicates the extra data associated with the event. The <u>UserData</u> property of the event indicates an extra data associated with the event. For instance, the LabelProperty = "<%=%256%> <%=%6%>" displays on the first line, the event's short margins, while on the second line displays the event's user data. Once you update or edit the event's UserData, the event's body automatically shows the new label. (Variant expression)
exEventDuration	7	Gets or sets the duration of the event. The returned values is of float type, and it indicates the duration of the event in days. For instance, the 1.5 indicates, 1 day and 12 hours. For instance, the LabelProperty = "< $%$ = $%256$ %> < $%$ = ((1:=int(0:= (date(%2)-date(%1)))) != 0? (=:1 + ' day(s)') : ") + (=:1?':") + ((1:=int(0:=((=:0 - =:1 + 1/24/60/60/2)*24))) != 0? =:1 + ' hour(s)': ") + (=:1?':") + ((1:=round((=:0 - =:1)*60)) != 0? =:1 + ' min(s)' : ")%>" displays on the first line, the event's short margins, while on the second line displays the event's durations in days, hours and minutes. Once you update or edit the event's margins, the event's body automatically shows the new length. You can use the MoveBy method to delay the current event with a specified value time. You can use the KnownProperty(exEventDuration) to change the event's duration. (Float expression)
		Specifies the repetitive expression of the event. The <u>Repetitive</u> property of the event indicates the expression that determines whether the event is repetitive. For instance, the LabelProperty = "

exEventRepetitiveExpression	8	<%=%256%> <%=len(%8)? `repetitive event`:``%>" displays repetitive event for repetitive events. (String expression)
exEventShortLabel	9	Specifies the short label of the event. This value returns the same value as <u>ShortLabel</u> property of the Event object. The ShortLabel property is shown if the event's body is too small. You can use this flag to associate the event's ShortLabel property to a field in the database, using the <u>DataField</u> property, when the <u>DataSource</u> property is used. (String Expression)
exEventLongLabel	10	Specifies the long label of the event. This value returns the same value as LongLabel property of the Event object. You can use this flag to associate the event's LongLabel property to a field in the database, using the DataField property, when the DataSource property is used. (String Expression)
exEventExtraLabel	11	Specifies the extra label of the event. This value returns the same value as <u>ExtraLabel</u> property of the Event object. You can use this flag to associate the event's ExtraLabel property to a field in the database, using the <u>DataField</u> property, when the <u>DataSource</u> property is used. (String Expression)
exEventID	12	Specifies the event's unique identifier. You can use this flag to associate the event's identifier property to a field in the database (usually the ID field in the table), using the <u>DataField</u> property, when the <u>DataSource</u> property is used. (Long expression)

Specifies the background color or the visual

exEventBodyBackColor	13	appearance of the event (body), equivalent of <u>BodyBackColor</u> property. (Long expression)
exEventBodyForeColor	14	Specifies the foreground color of the event (body), equivalent of <u>BodyForeColor</u> property. (Long expression)
exEventBodyPattern	15	Specifies the pattern of the event (body), equivalent of <u>BodyPattern</u> property. (String expression)
exEventShowStatus	16	Specifies whether the current event shows or hides its status, equivalent of <u>ShowStatus</u> property. (Boolean expression)
exEventStatusColor	17	Specifies the color of the event's status, equivalent of <u>StatusColor</u> property. (Long expression)
exEventStatusPattern	18	Specifies the pattern to show the event's status, equivalent of <u>StatusPattern</u> property. (String expression)
exEventLabelAlign	19	Indicates the alignment of the event's long label, equivalent of <u>LabelAlign</u> property. (<u>ContentAlignmentEnum</u> expression)
exEventExtraLabelAlign	20	Indicates the alignment of the event's extra label, equivalent of <u>ExtraLabelAlign</u> property. (<u>ContentAlignmentEnum</u> expression)
		Indicates the tooltip to be shown when the cursor hovers the event, equivalent of ToolTip property.

exEventToolTip	21	(String expression)
exEventToolTipTitle	22	Indicates the title of the tooltip to be shown when the cursor hovers the event, equivalent of <u>ToolTipTitle</u> property. (String expression)
exEventPictures	23	Specifies the list of pictures to be displayed on the event, equivalent of <u>Pictures</u> property. (String expression)
exEventPicturesAlign	24	Indicates the alignment of the event's picture, equivalent of <u>PicturesAlign</u> property. (<u>ContentAlignmentEnum</u> expression)
exEventExtraPictures	25	Specifies the list of extra pictures to be displayed on the event, equivalent of <u>ExtraPictures</u> property. (String expression)
exEventExtraPicturesAlign	26	Indicates the alignment of the event's extra picture, equivalent of <u>ExtraPicturesAlign</u> property. (<u>ContentAlignmentEnum</u> expression)
exEventEditable	27	Specifies whether the event's caption is editable, equivalent of <u>Editable</u> property. (<u>EditableCaptionEnum</u> expression)
exEventBodyBackgroundExt	28	Indicates additional colors, text, images that can be displayed on the event's background using the EBN string format, equivalent of <u>BodyBackgroundExt</u> property. (String expression)
EXEVENTMAX	29	Holds the count of properties (for internal use only)

exEventDisplayShortMargins	256	Displays the margins of the event in a short format (read-only). The <u>ShortDateFormat</u> property defines the short date format. The <u>ShortTimeFormat</u> property defines the short time format. (String expression)
exEventDisplayLongMargins	257	Displays the margins of the event in a long format (read-only). The LongDateFormat property defines the long date format. The LongTimeFormat property defines the long time format. (String expression)
exEventStartDate	258	Gets the starting date (not including the time) of the current event (read-only). You can use this property to get the starting date of the event. (DATE expression)
exEventStartTime	259	Gets the starting time (not including the date) of the current event (read-only). You can use this property to get the starting time of the event. (DATE expression, a value from 0 to 1)
exEventEndDate	260	Gets the ending date (not including the time) of the current event (read-only). You can use this property to get the ending date of the event. (DATE expression)
exEventEndTime	261	Gets the ending time (not including the date) of the current event (read-only). You can use this property to get the ending time of the event. (DATE expression, a value from 0 to 1)
		Gets the label of the owner group (read-only). The exEventGroupLabel property indicates the <u>Caption</u> property of the Group's event. The <u>GroupID</u> property of the event indicates the identifier of the group that event belongs to. The exEventGroupTitle

exEventGroupLabel	262	property indicates the <u>Title</u> property of the Group's event. For instance, the LabelProperty = " <%=%262%> <%=%256%>" displays on the first line, the group's caption, and the short margins of the event on the second line. The caption of the label is automatically updated once an event is moved from a group to another. (String expression)
exEventGroupTitle	263	Gets the title of the owner group (read-only). The exEventGroupTitle property indicates the <u>Title</u> property of the Group's event. The exEventGroupLabel property indicates the <u>Caption</u> property of the Group's event. The <u>GroupID</u> property of the event indicates the identifier of the group that event belongs to. For instance, the LabelProperty = "< $%$ = $%263$ %> < $%$ = $%256$ %>" displays on the first line, the group's caption, and the short margins of the event on the second line. The caption of the label is automatically updated once an event is moved from a group to another. (String expression)
exEventRepetitive	264	Indicates if the current event is a repetitive event. (read-only). You can use this flag to specify whether <u>Repetitive</u> property is not empty, and valid. (Boolean expression)
exEventDataSourceBookma	ırk 265	exEventDataSourceBookmark. Indicates the bookmark of the record associated with the current event (DataSource, read-only).

constants EventResizableEnum

The EventResizableEnum type indicates the margins of the events that can be resized. The <u>Resizable</u> event property indicates the margins of the event that user can resize at runtime. The <u>AllowResizeEvent</u> property indicates the combination of the keys to let user resizes the events. The <u>AllowMoveEvent</u> property indicates the combination of the keys to let user moves the events. The <u>Selectable</u> property specifies whether the event can be selected at runtime. The <u>Movable</u> property specifies whether the event can be moved at runtime.

The EventResizableEnum expression supports the following values:

Name	Value	e Description
exNoResizable	0	The event can not be resized.
exResizableStart	1	Only the starting point of the event can be resized.
exResizableEnd	2	Only the ending point of the event can be resized.
exResizableBoth	3	The event is sizable.

constants exClipboardFormatEnum

Defines the clipboard format constants. Use <u>GetFormat</u> property to check whether the clipboard data is of given type

Name	Value	Description
exCFText	1	Null-terminated, plain ANSI text in a global memory bloc.
exCFBitmap	2	A bitmap compatible with Windows 2.x.
exCFMetafile	3	A Windows metafile with some additional information about how the metafile should be displayed.
exCFDIB	8	A global memory block containing a Windows device-independent bitmap (DIB).
exCFPalette	9	A color-palette handle.
exCFEMetafile	14	A Windows enhanced metafile.
exCFFiles	15	A collection of files. Use <u>Files</u> property to get or set the collection of files.
exCFRTF	-1663	9A RTF document.

constants exOLEDragOverEnum

State transition constants for the OLEDragOver event

Name	Value	e Description
exOLEDragEnter	0	Source component is being dragged within the range of a target.
exOLEDragLeave	1	Source component is being dragged out of the range of a target.
exOLEDragOver	2	Source component has moved from one position in the target to another.

constants exOLEDropEffectEnum

Drop effect constants for OLE drag and drop events.

Name	Valu	e Description
exOLEDropEffectNone	0	Drop target cannot accept the data, or the drop operation was cancelled.
exOLEDropEffectCopy	1	Drop results in a copy of data from the source to the target. The original data is unaltered by the drag operation.
exOLEDropEffectMove	2	Drop results in data being moved from drag source to drop source. The drag source should remove the data from itself after the move.
exOLEDropEffectScroll	-214	7483643he is not implemented.

constants exOLEDropModeEnum

Constants for the OLEDropMode property, that defines how the control accepts OLE drag and drop operations. Use the <u>OLEDropMode</u> property to set how the component handles drop operations.

Name	Value	e Description
exOLEDropNone	0	The control is not used OLE drag and drop functionality.
exOLEDropManual	1	The control triggers the OLE drop events, allowing the programmer to handle the OLE drop operation in code.

Here's the list of events related to OLE drag and drop: <u>OLECompleteDrag</u>, <u>OLEDragDrop</u>, <u>OLEDragOver</u>, <u>OLEGiveFeedback</u>, <u>OLESetData</u>, <u>OLEStartDrag</u>.

constants LayoutChangingEnum

Generally, the control fires the <u>LayoutStartChanging</u> / <u>LayoutEndChanging</u> event when an UI operation is performed. The Operation parameter of the LayoutStartChanging / LayoutEndChanging events support the values as listed:

Name	Value	Description
exLayoutResizePanels	0	One of the panels has been resized. The <u>PaneWidth</u> property indicates the width of the left/right panel.
exCalendarSelectionChange	1	Specifies whether the selection in the calendar panel is changing/changed. The <u>Selection</u> property of the Calendar returns a safe array of selected dates. <i>The /NET or /WPF version provides the</i> <i>SelDates property of List<datetime> type to get or</datetime></i> <i>sets the new selection using a collection of</i> <i>DateTime objects.</i> The LayoutStartChanging(exCalendarSelectionChange) specifies that the user is about to change the selection in the calendar panel, while the LayoutEndChanging(exCalendarSelectionChange) specifies whether the user changed the selection in the calendar panel.
exCalendarFocusDateChange	e2	Specifies whether the focused date in the calendar panel is changing/changed. The FocusDate property indicates the date being focused in the calendar. The LayoutStartChanging(exCalendarFocusDateChange) specifies that a new date is about to be focused on the calendar panel, while the LayoutEndChanging(exCalendarFocusDateChange) specifies whether a new date has been focused.
exCalendarDateChange	3	Specifies whether the browsing date in the calendar panel is changing/changed. The <u>Date</u> property of the Calendar object indicates the month date being browsed in the calendar. The LayoutStartChanging(exCalendarDateChange) specifies that a new month is about to be shown on the calendar panel, while the LayoutEndChanging(exCalendarDateChange) specifies whether the new month has been browsed. The <u>FirstVisibleDate/LastVisibleDate</u> property indicates

		the first visible date in the calendar panel.
exScheduleMove	4	Specifies whether the control is automatically scrolled by drag and drop. By default, you can press the SHIFT + Click and drag the schedule view to a new position. The <u>AllowMoveSchedule</u> property allows the user to move or navigate the schedule view to a new position, without selecting a new date in the calendar panel.
exScheduleResize	5	The user is zooming the dates in the control. By default, you can click the middle mouse button, and drag the cursor to a new position, so the schedule view gets zoomed or resized. The <u>AllowResizeSchedule</u> property indicates the keys combination so the user can resize the schedule view at runtime.
exScheduleResizeTimeScale	6	The user is resizing the time scale. The <u>TimeScales</u> property access the control's <u>TimeScale</u> objects. The <u>Width</u> property of the time scale indicates the width in pixels of the time scale. The <u>AllowResizeTimeScale</u> property indicates the keys the user can resize at runtime the time scale.
exLayoutCalendarAutoHide	7	The calendar is auto shown or hidden. The OnResizeControl property on exCalendarAutoHide makes the calendar goes away if the cursor is not in it. The PaneWidth property indicates the width of the left/right panel. For instance, the PaneWidth(False) on 0, indicates that the calendar panel is hidden, or if it not zero, the calendar panel is shown. You can call the FitSelToView method during this operation so the schedule fits the selected dates in its client area.
exScheduleCreateEvent	8	The user crates a new event by dragging the mouse. The <u>AddEvent</u> event notifies your application once a new event is added to the schedule view.
exScheduleResizeGroup	9	The user resizes a group. The user can resize a group by clicking the groups header between two groups, and start dragging the cursor to a new position, and so the group is being resized. The <u>AllowResizeGroup</u> property specifies whether the user can resize a group at runtime.

exScheduleSelectionChange	10	Specifies whether the selection in the schedule panel is changing/changed. You can use the <u>AllowSelectEvent</u> property to change the key to allow the user select new events or you can prevent selecting any event using exDisallow value. The <u>Selectable</u> property of the event indicates whether the event can be selected at runtime. The <u>Selection</u> property gets or sets a safe array of selected events. <i>The /NET or /WPF version provides the</i> <i>SelEvents property of List<event> type to get or</event></i> <i>sets the new selection using a collection of Event</i> <i>objects.</i>
exScheduleMoveEvent	11	Indicates whether the user moves the event. The control fires the <u>UpdateEvent</u> event once the margin of the events are being updated.
exScheduleResizeStartEvent	12	Indicates whether the user resizes the start of the event. The control fires the <u>UpdateEvent</u> event once the margin of the events are being updated.
exScheduleResizeEndEvent	13	Indicates whether the user resizes the end of the event. The control fires the <u>UpdateEvent</u> event once the margin of the events are being updated.
exScheduleMoveMarkTime	14	Indicates whether the user is about to move a <u>MarkTime</u> object. The Movable property of the MarkTime object indicates whether the user can move at runtime the MarkTime object using the Mouse. The <u>MarkTimeFromPoint(-1,-1)</u> property indicates the <u>MarkTimeFromPoint(-1,-1)</u> property indicates the <u>MarkTimeFromPoint(-1,-1)</u> method during the LayoutStartChanging(exScheduleMoveMarkTime) to store the timer from the cursor to a global member, and when LayoutEndChanging(exScheduleMoveMarkTime) occurs, you can use the previously stored member to identify the timer being moved/updated at runtime.
		The user edit the event's caption. The event notifies once the user starts inline editing an appointment. The <u>Editable</u> property of the Event indicates the property of the Event to be edited at runtime. You can use the EventFromPoint(-1,-1) method during

exScheduleEditEvent	15	the LayoutStartChanging(exScheduleEditEvent) to store the event from the cursor to a global member, and when LayoutEndChanging(exScheduleEditEvent) occurs, you can use the previously stored member to identify the event being edited.
exLayoutExchangePanels	16	The panels of the control has been exchanged. The <u>AllowExchangePanels</u> property indicates the combination of keys that user can use so it can drag a panel from a position to another. The control provides two panels, the calendar panel and the schedule panel. By default, the calendar panel is displayed on the left side, while the schedule view is displayed on the right side. The <u>OnResizeControl</u> property specifies when the calendar/schedule view is displayed on left/right side of the control.
exScheduleScrollAllDayEvent	17	The user scrolls the all-day events. The <u>AllowAllDayEventScroll</u> property gets or sets a value that specifies whether the all-day event header supports scrolling.
exScheduleMoveGroup	18	The user moves a group. The <u>AllowMoveGroup</u> property specifies whether the user can move a group at runtime.
exUndo	33	An Undo operation is performed. The <u>AllowUndoRedo</u> property enables or disables the control's Undo/Redo feature.
exRedo	34	A Redo operation is performed. The <u>AllowUndoRedo</u> property enables or disables the control's Undo/Redo feature.
exUndoRedoUpdate	32	The Undo/Redo queue is updated. The <u>AllowUndoRedo</u> property enables or disables the control's Undo/Redo feature.

constants LinesStyleEnum

The LinesStyleEnum type specifies the type of lines the control can show. For instance, the <u>BorderSelStyle</u> property indicates the border to be shown in the schedule panel, around the selected **dates**. The <u>SelectEventStyle</u> property specifies whether the control should show lines to indicate the selected **events**. The LinesStyleEnum type supports the following values:

Name	Value	Description
exNoLines	-1	No lines are shown.
exLinesDot	0	The lines shows as dotted.
exLinesHDot4	1	The horizontal lines shows dotted.
exLinesVDot4	2	The vertical lines are shown as dotted.
exLinesDot4	3	The lines are shown as solid.
exLinesHDash	4	The horizontal lines are shown as dashed.
exLinesVDash	8	The vertical lines are shown as dashed.
exLinesDash	12	The lines are shown as dashed.
exLinesHSolid	16	The horizontal lines are shown as solid.
exLinesVSolid	32	The vertical lines are shown as solid.
exLinesSolid	48	The lines are shown as solid.
exLinesThick	256	The lines are shown ticker. This flag can be combined with any other flags, so the line is shown ticker.
exLinesThicker	768	The lines are shown ticker. This flag can be combined with any other flags, so the line is shown ticker.

constants IndexExtEnum

The IndexExtEnum type specifies the index of the part of the EBN object to be accessed. The Index parameter of the <u>BodyBackgroundExtValue</u> property indicates the index of the part of the EBN object to be changed or accessed. *The Exontrol's <u>eXButton</u> 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 BodyBackgroundExt property:



After calling the BodyBackgroundExt 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:

I

Name	Value	e 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 OnResizeControlEnum

The OnResizeControlEnum type indicates the options you have to specify what the control does when the control or a portion of the control is resized. The <u>OnResizeControl</u> property specifies the operation that the control performs when the user resizes the component.

You can use the OnResizeControl property to specify one of the followings:

- **auto hide** the calendar panel. Ability to hide the calendar section while the cursor is not in it (OnResizeControlEnum.exResizePanelRight Or OnResizeControlEnum.exCalendarFit Or OnResizeControlEnum.exCalendarAutoHide).
- hide completely the calendar section (exHideSplitter)
- specify the **alignment** of the calendar, as on the left or right side of the schedule view (OnResizeControlEnum.exChangePanels Or OnResizeControlEnum.exCalendarFit)
- full or partially view of the calendar panel (exResizePanelRight)
- disabling the control's vertical split bar (so user can not resize the fixed panel) (OnResizeControlEnum.exResizePanelRight Or OnResizeControlEnum.exDisableSplitter Or OnResizeControlEnum.exCalendarFit)

Name	Value	Description
exResizePanelLeft	0	Resizes the left panel of the control. If this flag is set the calendar view is resized once the control is resized, unless the exChangePanels flag is set, as the schedule will be resized.
exResizePanelRight	1	Resizes the right panel of the control. If this flag is set the schedule view is resized once the control is resized, unless the exChangePanels flag is set, as the calendar will be resized.
exDisableSplitter	128	Disables the splitter. The user can not resize the panels using the control's vertical split bar.
exHideSplitter	256	Hides the splitter. This flag allows you to display a single panel, the calendar or the schedule view at once. If the exHideSplitter is used in combination with the exChangePanels, the schedule view is shown only, else the calendar panel is displayed only/

The OnResizeControlEnum type supports the following values:

exChangePanels	512	Exchanges the content of the panels. If this flag is present, the schedule view is displayed on the left, while the calendar panel is shown on the right side of the component.
exCalendarFit	1024	Ensures that the calendar fits to the panel that hosts it. If this flag is present, the Calendar panel can not show its content partially.
exCalendarAutoHide	2048	Turns on or off the calendar panel when cursor leaves the panels. The auto hide feature allows you to hide the calendar panel, while the cursor is not in it, so the schedule view gains the entire client area.

constants OnSelectDateEnum

The OnSelectDateEnum type specifies the action the control performs once a new date is selected in the calendar panel. The <u>OnSelectDate</u> property indicates the operation to perform when user selects a new date in the calendar panel. The OnSelectDateEnum type supports the following values:

Name	Value	e Description
exFitSelToView	-1	The schedule view adjusts its size and position, so all selected dates are visible in the view.
exNoViewChange	0	The control does nothing once the user selects new dates in the calendar panel.
exEnsureVisibleDate	1	Ensures that the selected date is visible on the schedule view without resizing the view.

constants PaddingEdgeEnum

The PaddingEdgeEnum type defines the left, top, right and bottom padding to display the text. The <u>DefaultEventPadding</u> property defines the padding to display the text in the event's body. The PaddingEdgeEnum type defines the following predefined values:

Name	Value	e Description
exPaddingAll	-1	Indicates all margins of the object.
exPaddingLeft	0	Indicates the left margin of the object.
exPaddingTop	1	Indicates the top margin of the object.
exPaddingRight	2	Indicates the right margin of the object.
exPaddingBottom	3	Indicates the bottom margin of the object.

constants PatternEnum

The PatternEnum type specifies the type of patterns that the element can fill with. The <u>Type</u> property indicates the pattern to fill the element. The <u>Color</u> property indicates the color to fill the element's pattern, while the <u>FrameColor</u> property indicates the color to show the element's border/frame if the Type property includes the exPatternFrame flag.

The PatternEnum type supports the following values:

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	
exPatternF2Diagonal	13	
exPatternB2Diagonal	14	·/////////////////////////////////////
exPatternFrame	256	TheexPatternFrame can be combined with any othervalue. The FrameColor property indicates the colorto show the frame.
exPatternFrameThick	768	E . The exPatternFrameThick can be combined with any other value. The FrameColor property indicates the color to show the frame.

constants PictureDisplayEnum

The PictureDisplayEnum type defines the way the control's Picture is arranged on the control. The <u>Picture</u> property assign a picture to be displayed on the control's background. The <u>PictureDisplay</u> property indicates how the picture is layout on the control's background. The <u>BackColor</u> property specifies a solid color to be shown on the control's background. The <u>Background</u>(exCalendarBackColor) property changes the calendar's panel backcolor if not-zero.

The PictureDisplayEnum type supports the following values:

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

constants ScrollBarEnum

The ScrollBarEnum type specifies the vertical or horizontal scroll bar in the control. Use the <u>ScrollBars</u> property to specify whether the vertical or horizontal scroll bar is visible or hidden. Use the ScrollPartVisible property to specify the visible parts in the control's scroll bars (this property may not be available yet).

Name	Value	Description
exVScroll	0	Indicates the vertical scroll bar.
exHScroll	1	Indicates the horizontal scroll bar.

constants ScrollBarsEnum

Specifies which scroll bars will be visible on a control. The <u>ScrollBars</u> property of the control specifies the scroll bars being visible in the control. By default, the ScrollBars property is exBoth, which indicates that both scroll bars of the component are being displayed only when they require

Name	Value	e Description
exNoScroll	0	No scroll bars are shown
exHorizontal	1	The horizontal scroll bar is shown, if it is necessary.
exVertical	2	The vertical scroll bar is shown, if it is necessary.
exBoth	3	(default) Both horizontal and vertical scroll bars are shown, if they are necessary.
exDisableNoHorizontal	5	The horizontal scroll bar is always shown, it is disabled if it is unnecessary.
exDisableNoVertical	10	The vertical scroll bar is always shown, it is disabled if it is unnecessary.
exDisableBoth	15	Both horizontal and vertical scroll bars are always shown, disabled if they are unnecessary.

constants ScrollPartEnum

The ScrollPartEnum type defines the parts in the control's scrollbar. Use the ScrollPartVisible property to specify the visible parts in the control's scroll bar. Use the ScrollPartCaption property to specify the caption being displayed in any part of the control's scrollbar. The control fires the ScrollButtonClick event when the user clicks any button in the control's scrollbar.



Name	Value	e Description
exExtentThumbPart	6553	6 The thumb-extension part.
exLeftB1Part	3276	⁸ (L1) The first additional button, in the left or top area. By default, this button is hidden.
exLeftB2Part	1638	⁴ (L2) The second additional button, in the left or top area. By default, this button is hidden.
exLeftB3Part	8192	(L3) The third additional button, in the left or top area. By default, this button is hidden.
exLeftB4Part	4096	(L4) The forth additional button, in the left or top area. By default, this button is hidden.
exLeftB5Part	2048	(L5) The fifth additional button, in the left or top area. By default, this button is hidden.
exLeftBPart	1024	(<) The left or top button. By default, this button is visible.
exLowerBackPart	512	The area between the left/top button and the thumb. By default, this part is visible.
exThumbPart	256	The thumb part or the scroll box region. By default, the thumb is visible.
exUpperBackPart	128	The area between the thumb and the right/bottom button. By default, this part is visible.
exBackgroundPart	640	The union between the exLowerBackPart and the exUpperBackPart parts. By default, this part is visible.
exRightBPart	64	(>) The right or down button. By default, this button is visible.

exRightB1Part	32	(R1) The first additional button in the right or down side. By default, this button is hidden.
exRightB2Part	16	(R2) The second additional button in the right or down side. By default, this button is hidden.
exRightB3Part	8	(R3) The third additional button in the right or down side. By default, this button is hidden.
exRightB4Part	4	(R4) The forth additional button in the right or down side. By default, this button is hidden
exRightB5Part	2	(R5) The fifth additional button in the right or down side. By default, this button is hidden.
exRightB6Part	1	(R6) The sixth additional button in the right or down side. By default, this button is hidden.
exPartNone	0	No part.

constants SelectCalendarDateEnum

The SelectCalendarDateEnum type specifies the type of selection the <u>Select</u> method can perform. The <u>Select</u> method selects programmatically the current year, month, week, week day or focused day in the calendar panel. The

LayoutEndChanging(exCalendarSelectionChange) event occurs once the Select method is invoked.

The SelectCalendarDateEnum t	ype supports the	following flags:

Name	Value	Description
exSelectNothing	0	The flag does nothing.
exSelectYear	1	The entire year is being selected. This flag can be combined with the exSelectToggle.
exSelectMonth	2	The current month is being selected. This flag can be combined with the exSelectToggle.
exSelectWeek	3	The current week is being selected. This flag can be combined with the exSelectToggle.
exSelectWeekDay	4	The current week days of the current month are selected. This flag can be combined with the exSelectToggle.
exSelectFocusDay	5	The current focused day is selected. This flag can be combined with the exSelectToggle.
exSelectToggle	16	The selection is toggled. For instance, the exSelectFocusDay Or exSelectToggle can select or unselect the focused date in the calendar panel. If the flag exSelectToggle is present, the current selection is not cleared. If the flag exSelectToggle is used, the current selection is cleared.

constants SelectCreateEventEnum

The <u>AllowSelectCreateEvent</u> property specifies whether the newly created event gets selected or highlighted. Use the <u>ShowAllDayHeader</u> property to show the schedule's All-Day header so all All-Day evens are shown on this header. The <u>AllowCreateAllDayEvent</u> property has effect only when the schedule displays no time scale. The <u>AllDayEvent</u> property indicates an all-day event. You can use the <u>AllowSelectEvent</u> property to change the key to allow the user select new events or you can prevent selecting any event using exDisallow value. The SelectCreateEventEnum type supports the following values:

Name	Value	e Description
exSelectCreateEventNone	0	No event is selected when it is first created.
exSelectCreateEvent	1	The event being created gets selected.
exHighlightCreateEvent	2	The event being created gets highlighted for s short period of time (1 second).

constants ShowEventsEnum

The ShowEventsEnum type indicates the events to be displayed in the control. The <u>ShowEvents</u> property indicates the type of the events which schedule displays. *For instance, the ShowEvents on 0 (zero), indicates no events are shown on the control. the ShowEvents on 2 (two), indicates that the schedule view displays the repetitive events only.* The <u>EventsTransparent</u> property indicates the transparency to show the events on the schedule view.

The ShowEventsEnum type supports one or a combination of the following flags:

Name	Value	Description
exShowAllEvents	-1	Shows all events.
exShowRegularEvents	1	Shows the regular events.
exShowRepetitiveEvents	2	Shows the repetitive events. The <u>KnownProperty</u> (exEventRepetitive) indicates whether the event is an repetitive event. The <u>Repetitive</u> property indicates the expression that defines the dates where the event occurs.

constants ShowHighlightDateEnum

The ShowHighlightDateEnum type specifies the way the control can highlight the date in the calendar and/or schedule panel. The <u>ShowHighlightDate</u> property indicates whether the control highlights the date in the calendar and/or schedule panel. The <u>ShowHighlightDate</u> property can be a combination of one or more of the following values. The <u>HighlightDate</u> property indicates the color(s) to highlight the date. The ShowHighlightDateEnum type supports the following values:

Name	Value	Description
exHideHighlightDate	0	No highlight for any date.
exShowHighlightDateCalenda	r1	The dates being highlighted are shown in the calendar panel.
exShowHighlightDateSchedule	e2	The dates being highlighted are shown in the schedule panel.
exShowHighlightDate	3	The dates being highlighted are shown in the calendar and schedule panel.
exHighlightDateCalendarVertic	c a 6	The colors to highlight a date are vertically displayed in the calendar panel.
exHighlightDateScheduleVertie	c a 2	The colors to highlight a date are vertically displayed in the schedule panel.
exHighlightDateVertical	48	The colors to highlight a date are vertically displayed.
exHighlightDateCalendarGrad	i 25 6	The colors to highlight a date are shown on gradient, for the calendar panel.
exHighlightDateScheduleGrad	11 511 2	The colors to highlight a date are shown on gradient, for the schedule panel.
exHighlightDateGradient	768	The colors to highlight a date are shown on gradient.
exHighlightDateCalendarEllipt	i ciOl9 6	Clips the highlight of the date in the calendar panel to an ellipse around the date.
exHighlightDateScheduleEllipt	i &C1 \$2	Clips the highlight of the date in the schedule panel to an ellipse around the date.
exHighlightDateEllipticClip	12288	Clips the highlight of the date to an ellipse around the date.
exHighlightGroupingEvents	65536	Highlights the date based on the grouping events.

constants ShowMarkZoneEnum

The ShowMarkZoneEnum type indicates how the time-zones are shown in the control. Use the <u>Add</u> method of the MarkZones collection to add a new time-zone to the control. The <u>MarkZoneFromPoint</u> property indicates the time-zone from the cursor. The <u>ShowMarkZone</u> property shows or hides the added time-zones. Using the ShowMarkZone property the mark zones can be shown:

- hidden, exHideMarkZones
- on the back of the other elements as events, and so on, exShowMarkZonesBack
- on the front of the other elements as events, and so on, exShowMarkZonesFront (by default)
- using a semi-transparent color, exShowMarkZonesSemi (by default)

The ShowMarkZoneEnum type supports the following values:

Name	Value	e Description
exHideMarkZones	0	No mark zone is shown.
exShowMarkZonesBack	1	The mark zones are shown on the background.
exShowMarkZonesFront	2	The mark zones are displayed on front.
exShowMarkZonesSemi	3	The schedule shows the mark zones using a semi- transarent color.

constants ShowNonworkingTimeEnum

The ShowNonworkingTimeEnumtype indicates the way the control can display the nonworking time intervals. The ShowNonworkingTime property shows or hides the defined nonworking intervals. The <u>Add</u> method of the NonworkingTimes objects adds a new nonworking time interval. The <u>NonworkingTimes</u> collection is accessible through the <u>NonworkingTimes</u> property of the control.

The ShowNonworkingTimeEnum type supports the following values:

Name	Value	Description
exHideNonworkingTime	0	The nonworking time of the date is not shown.
exShowNonworkingTimeBack	1	The nonworking time of the date is shown on the background.
exShowNonworkingTimeFront	2	The nonworking time of the date is displayed on front.
exShowNonworkingTimeSemi	3	The nonworking time of the date is displayed using a semi-transparent color.

constants ShowViewCompactEnum

The ShowViewCompactEnum type indicates the way the control can show dates in the schedule panel. The <u>ShowViewCompact</u> property specifies the way the control arranges the dates in the schedule view.

Name	Value	Description
exViewCalendar	0	Default. The schedule view arranges the days as they are shown in the calendar panel.
exViewCalendarCompact	-1	The schedule view arranges the days as they are shown in the calendar panel, excepts that the first day of the month starts right after the last day of the previously month, or start to a new row.
exViewSingleRow	1	The schedule view arranges all days to a single row (horizontally).
exViewSingleColumn	2	The schedule view arranges all days to a single column (vertically).
exViewSingleRowLockHeade	er3	The schedule view arranges all days to a single row (horizontally), while the date header is shown/locked on the top while the chart is vertically scrolled.

constants UIVisualThemeEnum

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

Name	Value Description	
exNoVisualTheme	0 exNoVisualTheme	
exDefaultVisualTheme	1677721xDefaultVisualTheme	
exButtonsVisualTheme	4 exButtonsVisualTheme	
exCalendarVisualTheme	8 exCalendarVisualTheme	
exCheckBoxVisualTheme	64 exCheckBoxVisualTheme	
constants WeekDayEnum

The WeekDayEnum type indicates the days in the week. The <u>WeekDays</u> property indicates the name of the days in the week. The WeekDayEnum type includes the following values.

Name	Valu	e Description
exSunday	0	Sunday
exMonday	1	Monday
exTuesday	2	Tuesday
exWednesday	3	Wednesday
exThursday	4	Thursday
exFriday	5	Friday
exSaturday	6	Saturday

constants WeekNumberAsEnum

The WeekNumberAsEnum type specifies the ways the control displays the week number for dates. The <u>ShowWeeks</u> property specifies whether the week number header is shown or hidden. The <u>DisplayWeekNumberAs</u> property specifies the way the control displays the week number. The <u>FirstWeekDay</u> 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 control provides the <u>VisualDesign</u> property that allows you to easily change the control's visual appearance at design mode. Also, the VisualDesign property can be used at runtime to specify a visual appearance, by setting the VisualDesign property with a new generated value. The <u>UseVisualTheme</u> property indicates whether the current visual theme is applied to parts of the control.

Here's a screen shot skins a few UI parts of the component, using the EBN objects :



The Appearance object supports the following properties and methods:

Name	Description
Add	Adds or replaces a skin object to the control.
<u>Clear</u>	Removes all skins in the control.
Remove	Removes a specific skin from the control.
<u>RenderType</u>	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.

Туре	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.
	A string expression that indicates:
	 an Windows XP Theme part, it should start with "XP:". 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. In this case the format of the Skin parameter should be: "XP: Control/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 like listed at the end of the document. This option is available only on Windows XP that supports Themes API. copy of another skin with different coordinates, if it begins with "CP:". For instance, you may need to display a specified skin on a smaller rectangle. In this case, the string starts with "CP:", and contains the following "CP:n I t r b", where the n is the identifier being copied, the I, t, r, and b indicate the left, top, right and bottom coordinates being used to adjust the rectangle where the skin is displayed. For instance, the "CP:1 4 0 -4 0", indicates that the skin is displayed on a smaller rectangle like follows. Let's say that the control requests painting the {10, 10, 30, 20} area, a rectangle with the width of 20 pixels, and
Skin as Variant	the height of 10 pixels, the skin will be displayed on the {14,10,26,20} as each coordinates in the "CP" syntax is added to the displayed rectangle, so the skin looks smaller. This way you can apply different effects to your objects in your control. The following screen shot shows the control's header when using a "CP:1 -6 -6 6 6", that displays the original skin on larger rectanges.



- the path to the skin file (*.ebn). The <u>Exontrol's</u> <u>exButton</u> component installs a skin builder that should be used to create new skins
- the BASE64 encoded string that holds a skin file (
 .ebn). Use the Exontrol's <u>exImages</u> tool to build BASE 64 encoded strings on the skin file (.ebn) you have created. Loading the skin from a file (eventually uncompressed file) is always faster then loading from a BASE64 encoded string

A byte[] or safe arrays of VT_I1 or VT_UI1 expression that indicates the content of the EBN file. You can use this 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)).

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 <u>Remove</u> method to remove a specific skin from the control. Use the <u>Clear</u> method to remove all skins in the control. Use the <u>BeginUpdate</u> and <u>EndUpdate</u> methods to maintain performance while init the control. Use the <u>Refresh</u> method to refresh the control.

The control provides the <u>VisualDesign</u> property that allows you to easily change the control's visual appearance at design mode. Also, the VisualDesign property can be used at runtime to specify a visual appearance, by setting the VisualDesign property with a new generated value. The <u>UseVisualTheme</u> property indicates whether the current visual theme is applied to parts of 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 &H200000 indicates that we apply the skin with the index 2 using the old color, to the object that BackColor is applied.

The following samples applies the EBN (<u>normal.ebn</u>) to all events.

VBA (MS Access, Excell...)

```
With Schedule1

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

.Background(75) = &H1000000

.Background(79) = &H1000000

.BodyEventBackColor = &H1000000

End With
```

VB6

```
With Schedule1
.VisualAppearance.Add 1,"c:\exontrol\images\normal.ebn"
```

```
.Background(exScheduleCreateEventBackColor) = &H1000000
.Background(exScheduleUpdateEventsBackColor) = &H1000000
.BodyEventBackColor = &H1000000
End With
```

VB.NET

```
With Exschedule1
.VisualAppearance.Add(1,"c:\exontrol\images\normal.ebn")
```

.set_Background32(exontrol.EXSCHEDULELib.BackgroundPartEnum.exScheduleCreateEv

.set_Background32(exontrol.EXSCHEDULELib.BackgroundPartEnum.exScheduleUpdatel

.BodyEventBackColor32 = &H1000000 End With

VB.NET for /COM

```
With AxSchedule1
```

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

.set_Background(EXSCHEDULELib.BackgroundPartEnum.exScheduleCreateEventBackCo

.set_Background(EXSCHEDULELib.BackgroundPartEnum.exScheduleUpdateEventsBack(

.GetOcx().**BodyEventBackColor** = &H1000000 End With

C++

Copy and paste the following directives to your header file as it defines the namespace 'EXSCHEDULELib' for the library: 'ExSchedule 1.0 Control Library'

```
#import <ExSchedule.dll>
using namespace EXSCHEDULELib;
*/
EXSCHEDULELib::ISchedulePtr spSchedule1 = GetDlgItem(IDC_SCHEDULE1)-
>GetControlUnknown();
spSchedule1->GetVisualAppearance()->Add(1,"c:\\exontrol\\images\\normal.ebn");
spSchedule1-
>PutBackground(EXSCHEDULELib::exScheduleCreateEventBackColor,0x100000);
spSchedule1-
>PutBackground(EXSCHEDULELib::exScheduleUpdateEventsBackColor,0x100000);
spSchedule1->PutBodyEventBackColor(0x100000);
```

C++ Builder

```
Schedule1->VisualAppearance-

> Add(1,TVariant("c:\\exontrol\\images\\normal.ebn"));

Schedule1-

> Background[Exschedulelib_tlb::BackgroundPartEnum::exScheduleCreateEventBackColo

= 0x1000000;

Schedule1-

> Background[Exschedulelib_tlb::BackgroundPartEnum::exScheduleUpdateEventsBackColo

= 0x1000000;
```

Schedule1->BodyEventBackColor = 0x100000;

C#

exschedule1.VisualAppearance.**Add**(1,"c:\\exontrol\\images\\normal.ebn"); exschedule1.set_Background32(exontrol.EXSCHEDULELib.BackgroundPartEnum.exSche

exschedule1.set_Background32(exontrol.EXSCHEDULELib.BackgroundPartEnum.exSche

exschedule1.BodyEventBackColor32 = 0x1000000;

JavaScript

```
<OBJECT classid="clsid:9B09E13D-7A88-4299-9DBE-383380435377"
id="Schedule1"></OBJECT>
<SCRIPT LANGUAGE="JScript">
Schedule1.VisualAppearance.Add(1,"c:\\exontrol\\images\\normal.ebn");
Schedule1.Background(75) = 16777216;
Schedule1.Background(79) = 16777216;
Schedule1.BodyEventBackColor = 16777216;
</SCRIPT>
```

C# for /COM

axSchedule1.VisualAppearance.**Add**(1,"c:\\exontrol\\images\\normal.ebn"); axSchedule1.set_Background(EXSCHEDULELib.BackgroundPartEnum.exScheduleCreatel

axSchedule1.set_Background(EXSCHEDULELib.BackgroundPartEnum.exScheduleUpdate

(axSchedule1.GetOcx() as EXSCHEDULELib.Schedule).**BodyEventBackColor** = 0x1000000;

X++ (Dynamics Ax 2009)

```
public void init()
{
;
super();
exschedule1.VisualAppearance().Add(1,"c:\\exontrol\\images\\normal.ebn");
exschedule1.Background(75/*exScheduleCreateEventBackColor*/,0x1000000);
exschedule1.Background(79/*exScheduleUpdateEventsBackColor*/,0x1000000);
exschedule1.BodyEventBackColor(0x1000000);
}
```

Delphi 8 (.NET only)

with AxSchedule1 do

```
begin
```

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

set_Background(EXSCHEDULELib.BackgroundPartEnum.exScheduleCreateEventBackCol

set_Background(EXSCHEDULELib.BackgroundPartEnum.exScheduleUpdateEventsBackC

(GetOcx() as EXSCHEDULELib.Schedule).**BodyEventBackColor** := \$1000000; end

Delphi (standard)

```
with Schedule1 do
begin
    VisualAppearance.Add(1,'c:\exontrol\images\normal.ebn');
    Background[EXSCHEDULELib_TLB.exScheduleCreateEventBackColor] := $1000000;
    Background[EXSCHEDULELib_TLB.exScheduleUpdateEventsBackColor] :=
$1000000;
BodyEventBackColor := $1000000;
end
```

VFP

```
with thisform.Schedule1
.VisualAppearance.Add(1,"c:\exontrol\images\normal.ebn")
.Object.Background(75) = 0x1000000
.Object.Background(79) = 0x1000000
.BodyEventBackColor = 0x1000000
endwith
```

dBASE Plus

```
local oSchedule
```

oSchedule = form.Activex1.nativeObject
oSchedule.VisualAppearance.Add(1,"c:\exontrol\images\normal.ebn")
oSchedule.Template = [Background(75) = 0x1000000] //

```
oSchedule.Background(75) = 0x1000000
oSchedule.Template = [Background(79) = 0x1000000] //
oSchedule.Background(79) = 0x1000000
oSchedule.BodyEventBackColor = 0x1000000
```

XBasic (Alpha Five)

```
Dim oSchedule as P

oSchedule = topparent:CONTROL_ACTIVEX1.activex

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

oSchedule.Template = "Background(75) = 16777216" ' oSchedule.Background(75)

= 16777216

oSchedule.Template = "Background(79) = 16777216" ' oSchedule.Background(79)

= 16777216

oSchedule.BodyEventBackColor = 16777216
```

Visual Objects

oDCOCX_Exontrol1:VisualAppearance:**Add**(1,"c:\exontrol\images\normal.ebn") oDCOCX_Exontrol1:[Background,exScheduleCreateEventBackColor] := 0x1000000 oDCOCX_Exontrol1:[Background,exScheduleUpdateEventsBackColor] := 0x1000000 oDCOCX_Exontrol1:**BodyEventBackColor** := 0x1000000

PowerBuilder

```
OleObject oSchedule
oSchedule = ole_1.Object
oSchedule.VisualAppearance.Add(1,"c:\exontrol\images\normal.ebn")
oSchedule.Background(75,16777216 /*0x1000000*/)
oSchedule.Background(79,16777216 /*0x1000000*/)
oSchedule.BodyEventBackColor = 16777216 /*0x1000000*/
```

On Windows XP, the following table shows how the common controls are broken into parts

and states:

Control/ClassName)	Part	States
			CBS_UNCHECKED 1 CBS_UNCHECKE CBS_UNCHECKED = 3
BUTTON	BP_CHECKBOX = 3		CBS_UNCHECKED = 4 CBS_CHECKEI 5 CBS_CHECKEDF CBS_CHECKEDPR CBS_CHECKEDDIS CBS_MIXEDNORM CBS_MIXEDHOT = CBS_MIXEDPRESS
	BP_GROUPBOX = 4		CBS_MIXEDDISAB GBS_NORMAL = 1 GBS_DISABLED =
	BP_PUSHBUTTON = 1		= 2 PBS_PRESSEI PBS_DISABLED = PBS_DEFAULTED = RBS_UNCHECKED 1 RBS_UNCHECKE
	BP_RADIOBUTTON =	2	RBS_UNCHECKED = 3 RBS_UNCHECKED = 4 RBS_CHECKED 5 RBS_CHECKED RBS_CHECKEDPR RBS_CHECKEDDIS
	BP USERBUTTON = 5	5	
CLOCK	CLP_TIME = 1		CLS_NORMAL = 1 CBXS_NORMAL =
COMBOBOX	CP_DROPDOWNBUT	TON = 1	CBXS_HOT = 2 CBXS_PRESSED = CBXS_DISABLED =
EDIT	EP_CARET = 2		_
			ETS_NORMAL = 1 2 ETS_SELECTED

	EP_EDITTEXT = 1	ETS_FOCUSED = { ETS_READONLY = ETS_ASSIST = 7
EXPLORERBAR	EBP_HEADERBACKGROUND = 1	
	EBP_HEADERCLOSE = 2	EBHC_NORMAL = EBHC_HOT = 2 EBHC_PRESSED =
	EBP_HEADERPIN = 3	EBHP_NORMAL = EBHP_HOT = 2 EBHP_PRESSED = EBHP_SELECTED 4 EBHP_SELECTEI EBHP_SELECTEDF 6
	EBP_IEBARMENU = 4	EBM_NORMAL = 1 = 2 EBM_PRESSEI
	EBP_NORMALGROUPBACKGROUND = 5	
	EBP_NORMALGROUPCOLLAPSE = 6	EBNGC_NORMAL EBNGC_HOT = 2 EBNGC_PRESSED
	EBP_NORMALGROUPEXPAND = 7	EBNGE_HOT = 2 EBNGE_PRESSED
	EBP_NORMALGROUPHEAD = 8	
	EBP_SPECIALGROUPBACKGROUND = 9	
	EBP_SPECIALGROUPCOLLAPSE = 10	EBSGC_NORMAL EBSGC_HOT = 2 EBSGC_PRESSED EBSGE_NORMAL
	EBP_SPECIALGROUPEXPAND = 11	EBSGE_HOT = 2 EBSGE_PRESSED
	EBP_SPECIALGROUPHEAD = 12	
HEADER	HP_HEADERITEM = 1	HIS_NORMAL = 1 2 HIS_PRESSED =
	HP_HEADERITEMLEFT = 2	HILS_NORMAL = 1 = 2 HILS_PRESSEI

HP_HEADERITEMRIGHT = 3

HP_HEADERSORTARROW = 4

HIRS_NORMAL = 1 = 2 HIRS_PRESSE

HSAS_SORTEDUP

LISTVIEW LVP_EMPTYTEXT = 5 LVP_LISTDETAIL = 3 LVP_LISTGROUP = 2 LVP_LISTITEM = 1 LVP_LISTITEM = 1 LVP_LISTSORTEDDETAIL = 4 MENU MP_MENUBARDROPDOWN = 4 MP_MENUBARDROPDOWN = 4 MP_MENUBARITEM = 3 MP_CHEVRON = 5 MP_CHEVRON = 5 MP_CHEVRON = 5 MP_MENUDROPDOWN = 2 MP_MENUDROPDOWN = 2 MP_MENUDROPDOWN = 2 MP_MENUTEM = 1 MP_MENUTEM = 1 MP_MENUTEM = 1 MP_MENUTEM = 1 MP_MENUTEM = 1 MP_SEPARATOR = 6 MS_SELECTED = : MS_DEMOTED = 3 MS_NORMAL = 1 MP_SEPERATOR = 6 MS_NORMAL = 1 MS_SELECTED = : MS_DEMOTED = 3 MS_NORMAL = 1 MP_MENUTEM = 1 MP_MENUTEM = 1 MP_SEPARATOR = 6 MS_SELECTED = : MS_DEMOTED = 3 MS_NORMAL = 1 MS_SELECTED = : MS_DEMOTED = 2 MS_NORMAL = 1 MP_SEPARATOR = 6 MS_SELECTED = : MS_NORMAL = 1 MS_SELECTED = : MS_DEMOTED = 3 MS_NORMAL = 1 MS_NORMAL = 1 MS_SELECTED = : MS_DEMOTED = 2 MS_NORMAL = 1 MS_NORMAL = 1 MS_SELECTED = : MS_DEMOTED = 2 MS_NORMAL = 1 MS_NORMAL =			HSAS SORTEDDC
LVP_LISTDETAIL = 3 LVP_LISTGROUP = 2 LVP_LISTGROUP = 2 LVP_LISTITEM = 1 LVP_LISTITEM = 1 LVP_LISTSORTEDDETAIL = 4 MENU MP_MENUBARDROPDOWN = 4 MS_SELECTED =: MP_MENUBARITEM = 3 MP_CHEVRON = 5 MS_SELECTED =: MP_CHEVRON = 5 MS_SELECTED =: MP_MENUDROPDOWN = 2 MP_MENUDROPDOWN = 2 MS_SELECTED =: MS_NORMAL = 1 MP_CHEVRON = 5 MS_SELECTED =: MS_NORMAL = 1 MS_SELECTED =: MS_DEMOTED = 3 MS_SELECTED =: MS_DEMOTED = 3 MS_NORMAL = 1 MP_MENUDROPDOWN = 2 MS_NORMAL = 1 MS_SELECTED =: MS_NORMAL = 1 MS_NORMAL = 1 MP_SEPARATOR = 6 MS_NORMAL = 1 MDP_NEWAPPBUTTON = 1 MDP_SEPERATOR = 2 MDS_NORMAL = 1 MDP_SEPERATOR = 2 MS_NORMAL = 1 MDS_NORMAL = 1 MDS_NORMAL = 1 MDS_NORA	LISTVIEW	LVP EMPTYTEXT = 5	_
LVP_LISTGROUP = 2 LVP_LISTGROUP = 2 LVP_LISTITEM = 1 LVP_LISTITEM = 1 LVP_LISTSORTEDDETAIL = 4 MENU MP_MENUBARDROPDOWN = 4 MP_MENUBARITEM = 3 MP_MENUBARITEM = 3 MP_CHEVRON = 5 MP_CHEVRON = 5 MP_MENUDROPDOWN = 2 MP_MENUDROPDOWN = 2 MP_MENUITEM = 1 MP_MENUITEM = 1 MP_MENUITEM = 1 MP_SEPARATOR = 6 MS_SELECTED = : MS_DEMOTED = 3 MS_NORMAL = 1 MS_SELECTED = : MS_DEMOTED = 3 MS_NORMAL = 1 MP_MENUITEM = 1 MP_SEPARATOR = 6 MS_SELECTED = : MS_DEMOTED = 3 MS_NORMAL = 1 MS_SELECTED = : MS_DEMOTED = 2 MS_NORMAL = 1 MS_SELECTED = : MS_DEMOTED = 3 MS_NORMAL = 1 MS_SELECTED = : MS_DEMOTED = 3 MS_NORMAL = 1 MS_SELECTED = : MS_DEMOTED = 2 MS_NORMAL = 1 MS_SELECTED = : MS_DEMOTED = 3 MS_NORMAL = 1 MS_SELECTED = : MS_SELECTED = : MS_NORMAL = 1 MS_SELECTED = : MS_SELECTED = :		LVP LISTDETAIL = 3	
LVP_LISTRICOP = 2 LVP_LISTREM = 1 LVP_LISTREM = 1 LVP_LISTREM = 1 LVP_LISTSORTEDDETAIL = 4 MENU MP_MENUBARDROPDOWN = 4 MP_MENUBARITEM = 3 MP_MENUBARITEM = 3 MP_CHEVRON = 5 MP_CHEVRON = 5 MP_MENUDROPDOWN = 2 MP_MENUDROPDOWN = 2 MP_MENUITEM = 1 MP_MENUITEM = 1 MP_MENUITEM = 1 MP_SEPARATOR = 6 MS_NORMAL = 1 MS_SELECTED = : MS_NORMAL = 1 MS_NORMAL = 1 MS_SELECTED = : MS_DEMOTED = 3 MS_NORMAL = 1 MS_SELECTED = : MS_DEMOTED = 3 MS_NORMAL = 1 MS_SELECTED = : MS_DEMOTED = 3 MS_NORMAL = 1 MP_MENUITEM = 1 MP_SEPARATOR = 6 MS_SELECTED = : MS_DEMOTED = 3 MS_NORMAL = 1 MDS_NORMAL = 1 MS_SELECTED = : MS_DEMOTED = 3 MS_NORMAL = 1 MS_SELECTED = : MS_DEMOTED = 3 MS_NORMAL = 1 MS_SELECTED = : MS_DEMOTED = 3 MS_NORMAL = 1 MS_SELECTED = : MS_DEMOTED = 2 MDS_NORMAL = 1 MDS_NORMAL = 1 MS_NORMAL = 1 MS_NORMAL = 1 MS_NORMAL = 1 MS_NORMAL = 1 MS_NORMAL = 1 MS_NORMAL = 1 MS_SELECTED = : MS_DEMOTED = 2 MDS_NORMAL = 1 MS_NORMAL = 1 MS_N			
LVP_LISTITEM = 1 LVP_LISTITEM = 1 LVP_LISTSORTEDDETAIL = 4 MENU MP_MENUBARDROPDOWN = 4 MS_NORMAL = 1 MP_MENUBARITEM = 3 MP_CHEVRON = 5 MP_CHEVRON = 5 MP_MENUDROPDOWN = 2 MP_MENUDROPDOWN = 2 MS_NORMAL = 1 MP_MENUDROPDOWN = 2 MS_NORMAL = 1 MP_MENUDROPDOWN = 2 MS_NORMAL = 1 MP_MENUTEM = 1 MP_MENUTEM = 1 MP_SEPARATOR = 6 MS_NORMAL = 1 MP_SEPERATOR = 6 MS_NORMAL = 1 MDP_SEPERATOR = 2 MS_NORMAL = 1 MDS_NORMAL = 1 MDP_SEPERATOR = 2 MS_NORMAL = 1 MS_NORMAL = 1 MDP_SEPERATOR = 2 MS_NORMAL = 1 MS_NORMAL = 1 MS_NORMAL = 1 MDP_SEPERATOR = 2 MS_NORMAL = 1 MS_NORMAL = 1 MS_NO		$EVF_EISTGROOF = 2$	
LVP_LISTITEM = 1 LVP_LISTITEM = 1 LVP_LISTSORTEDDETAIL = 4 MENU MP_MENUBARDROPDOWN = 4 MP_MENUBARITEM = 3 MP_MENUBARITEM = 3 MP_CHEVRON = 5 MP_CHEVRON = 5 MP_MENUDROPDOWN = 2 MP_MENUDROPDOWN = 2 MP_MENUITEM = 1 MP_MENUITEM = 1 MP_SEPARATOR = 6 MS_NORMAL = 1 MS_SELECTED = : MS_NORMAL = 1 MS_SIZECTED = : MS_NORMAL = 1 MS_SIZECTED = : MS_NORMAL = 1 MS_SIZECTED = : MS_NORMAL = 1 = 2 MDS_NORMAL = 1 = 2 MDS_NORMAL = 1 = 2 DNS_NORMAL = 1 = 2 DNS_NORMAL = 1 MDS_SIZED = DNH_ZS_NORMAL = 1 = 2 DNS_NORMAL = 1 MDS_NORMAL = 1 MS_SIZECTED = : MDS_NORMAL = 1 MS_SIZECTED = : MS_NORMAL = 1 MS_SIZECTED = : MS_SIZECTED = : MS_SIZECTED = : MS_SIZECTED = : MS			2 LIS SELECTED :
LVP_LISTSORTEDDETAIL = 4 MENU MP_MENUBARDROPDOWN = 4 MP_MENUBARITEM = 3 MP_MENUBARITEM = 3 MP_CHEVRON = 5 MP_CHEVRON = 5 MP_MENUDROPDOWN = 2 MP_MENUDROPDOWN = 2 MP_MENUDROPDOWN = 2 MP_MENUITEM = 1 MP_SEPARATOR = 6 MP_SEPERATOR = 6 MENUBAND MDP_NEWAPPBUTTON = 1 MDP_SEPERATOR = 2 MDP_SEPERATOR = 2 PAGE PGRP_DOWN = 2 NDS_NORMAL = 1 MDS_DEMOTED = 3 MDS_NORMAL = 1 MS_SELECTED = ; MS_DEMOTED = 3 MS_NORMAL = 1 MS_DEMOTED = 3 MS_NORMAL = 1 MS_DEMOTED = 3 MS_NORMAL = 1 MS_SELECTED = ; MS_NORMAL = 1 MS_SELECTED = ; MS_SORMAL = 1 MS_SELECTED = ; MS_SORMAL = 1 MS_SELECTED = ; MS_SORMAL = 1 MS_SORMAL = 1 M		IVP LISTITEM = 1	$LIS_DISABLED = 4$
5 LVP_LISTSORTEDDETAIL = 4 MENU MP_MENUBARDROPDOWN = 4 MS_NORMAL = 1 MP_MENUBARITEM = 3 MS_SELECTED = : MP_MENUBARITEM = 3 MS_SELECTED = : MP_CHEVRON = 5 MS_SELECTED = : MP_MENUDROPDOWN = 2 MS_SELECTED = : MP_MENUDROPDOWN = 2 MS_SELECTED = : MP_MENUDROPDOWN = 2 MS_SELECTED = : MP_MENUITEM = 1 MS_SELECTED = : MP_SEPARATOR = 6 MS_SELECTED = : MENUBAND MDP_NEWAPPBUTTON = 1 MDP_SEPERATOR = 2 DNS_NORMAL = 1 PAGE PGRP_DOWN = 2 DNS_NORMAL = 1 PAGE PGRP_DOWN = 2 DNS_NORMAL = 1			LIS SELECTEDNO
LVP_LISTSORTEDDETAIL = 4 MENU MP_MENUBARDROPDOWN = 4 MP_MENUBARITEM = 3 MP_MENUBARITEM = 3 MP_CHEVRON = 5 MP_CHEVRON = 5 MP_MENUDROPDOWN = 2 MP_MENUDROPDOWN = 2 MP_MENUITEM = 1 MP_MENUITEM = 1 MP_SEPARATOR = 6 MS_SELECTED = : MS_DEMOTED = 3 MS_NORMAL = 1 MS_SELECTED = : MS_DEMOTED = 3 MS_NORMAL = 1 MS_DEMOTED = 3 MS_NORMAL = 1 MS_DEMOTED = 3 MS_NORMAL = 1 MS_DEMOTED = 3 MS_NORMAL = 1 MS_DEMOTED = 3 MS_NORMAL = 1 MS_DISABLED = MS_SELECTED = : MS_DEMOTED = 2 MDS_NORMAL = 1 MDS_DISABLED = MDS_CHECKED = MDS_HOTCHECKE MDP_SEPERATOR = 2 PAGE PGRP_DOWN = 2 PAGE PGRP_DOWN = 2 MS_NORMAL = 1 MS_SISABLED = DNS_NORMAL = 1 MDS_NORMAL = 1			5
MENU MP_MENUBARDROPDOWN = 4 MENU MP_MENUBARDROPDOWN = 4 MP_MENUBARITEM = 3 MP_MENUBARITEM = 3 MP_CHEVRON = 5 MS_DEMOTED = 3 MS_DEMOTED = 3 MS_DEMOTED = 3 MS_NORMAL = 1 MP_CHEVRON = 5 MS_DEMOTED = 3 MS_NORMAL = 1 MP_MENUDROPDOWN = 2 MS_DEMOTED = 3 MS_NORMAL = 1 MP_MENUITEM = 1 MP_MENUITEM = 1 MP_SEPARATOR = 6 MS_NORMAL = 1 MS_SELECTED = 3 MS_NORMAL = 1 MS_NORMAL = 1 = 2 MS_NORMAL = 1 MDS_DISABLED = 3 MDS_NORMAL = 1 = 2 MS_NORMAL = 1 = 2 MS_PRESSE MDS_NORMAL = 1 = 2 DNS_NORMAL = 1 = 2 DNS_NORMAL = 1 = 2 DNS_NORMAL = 1 DNHZS_NORMAL = 1 = 2 DNS_NORMAL = 1 DNHZS_NORMAL = 1 PAGE PGRP_DOWN = 2 PAGE PGRP_DOWN = 2 PGRP_DOWN		LVP LISTSORTEDDETAIL = 4	
MENUMP_MENUBARDROPDOWN = 4MS_SELECTED = : MS_DEMOTED = 3MP_MENUBARITEM = 3MS_NORMAL = 1MP_MENUBARITEM = 3MS_SELECTED = : MS_DEMOTED = 3MP_CHEVRON = 5MS_NORMAL = 1MP_MENUDROPDOWN = 2MS_SELECTED = : MS_DEMOTED = 3MP_MENUDROPDOWN = 2MS_SELECTED = : MS_DEMOTED = 3MP_MENUITEM = 1MS_SELECTED = : MS_SELECTED = : MS_DEMOTED = 3MP_SEPARATOR = 6MS_NORMAL = 1 MS_SELECTED = : MS_DEMOTED = 3MENUBANDMDP_NEWAPPBUTTON = 1MDS_DISABLED = MDS_NORMAL = 1 E 2 MDS_PRESSEI MDS_HOTCHECKEMENUBANDMDP_NEWAPPBUTTON = 1MDS_DISABLED = MDS_HOTCHECKEPAGEPGRP_DOWN = 2DNS_NORMAL = 1 = 2 DNS_PRESSEI DNS_DISABLED = DNHZS_NORMAL = 1 = 2 DNS_PRESSEI DNS_DISABLED = DNHZS_NORMAL = 1 DNHZS_NORMAL = 2		—	MS NORMAL = 1
MS_DEMOTED = 3 MP_MENUBARITEM = 3 MP_MENUBARITEM = 3 MP_CHEVRON = 5 MP_CHEVRON = 5 MP_MENUDROPDOWN = 2 MP_MENUDROPDOWN = 2 MP_MENUITEM = 1 MP_MENUITEM = 1 MP_SEPARATOR = 6 MS_SELECTED = ; MS_DEMOTED = 3 MS_NORMAL = 1 MS_SELECTED = ; MS_DEMOTED = 3 MS_NORMAL = 1 = 2 MDS_PRESSE MDS_NORMAL = 1 = 2 MDS_PRESSE MDS_ORMAL = 1 = 2 MDS_PRESSE MDS_ORMAL = 1 = 2 MDS_PRESSE MDS_ORMAL = 1 = 2 MDS_PRESSE MDS_ORMAL = 1 = 2 MDS_PRESSE MDS_NORMAL = 1 = 2 MDS_PRESSE MDS_NORMAL = 1 = 2 MDS_PRESSE MDS_NORMAL = 1 = 2 MDS_PRESSE MDS_ORMAL = 1 = 2 MDS_ORMAL = 1 = 2 MD	MENU	$MP_MENUBARDROPDOWN = 4$	MS_SELECTED = :
MP_MENUBARITEM = 3 MP_MENUBARITEM = 3 MP_CHEVRON = 5 MP_CHEVRON = 5 MP_MENUDROPDOWN = 2 MP_MENUDROPDOWN = 2 MP_MENUITEM = 1 MP_MENUITEM = 1 MP_SEPARATOR = 6 MS_NORMAL = 1 MS_SELECTED = ; MS_DEMOTED = 3 MS_NORMAL = 1 = 2 MDS_PRESSEI MDS_NORMAL = 1 = 2 MDS_PRESSEI MDS_OISABLED = MDS_HOTCHECKE MDP_SEPERATOR = 2 PAGE PGRP_DOWN = 2 PAGE PGRP_DOWN = 2 MS_NORMAL = 1 = 2 MDS_PRESSEI DNS_NORMAL = 1 = 2 MDS_PRESSEI DNS_NORMAL = 1 = 2 MDS_PRESSEI DNS_NORMAL = 1 = 2 DNS_PRESSEI DNS_NORMAL = 1 = 2 DNS_PRESSEI DNS_NORMAL = 1 = 2 DNS_PRESSEI DNS_DISABLED = DNHZS_NORMAL = 1 = 2 DNS_PRESSEI DNS_DISABLED = DNS_DISABLED = DNS_DISABLED = DNS_DISABLED = DNS_DISABLED = DNS_DISABLED = DNS_DISABLED = D			MS_DEMOTED = 3
MP_MENUBARITEM = 3 MP_MENUBARITEM = 3 MP_CHEVRON = 5 MP_CHEVRON = 5 MP_MENUDROPDOWN = 2 MP_MENUDROPDOWN = 2 MP_MENUITEM = 1 MP_MENUITEM = 1 MP_SEPARATOR = 6 MS_NORMAL = 1 MS_SELECTED = ; MS_DEMOTED = 3 MS_NORMAL = 1 MS_DEMOTED = 3 MS_NORMAL = 1 = 2 MDS_PRESSEI MDS_HOTCHECKE MDP_SEPERATOR = 2 PAGE PGRP_DOWN = 2 PAGE PGRP_DOWN = 2 NS_NORMAL = 1 MS_NORMAL = 1 MS_DISABLED = DNS_NORMAL = 1 MS_NORMAL =			MS_NORMAL = 1
MS_DEMOTED = 3 MS_NORMAL = 1 MP_CHEVRON = 5 MS_SELECTED = ; MS_DEMOTED = 3 MS_NORMAL = 1 MP_MENUDROPDOWN = 2 MS_SELECTED = ; MS_DEMOTED = 3 MS_NORMAL = 1 MP_MENUITEM = 1 MP_SEPARATOR = 6 MS_NORMAL = 1 MS_SELECTED = ; MS_DEMOTED = 3 MS_NORMAL = 1 MS_SELECTED = ; MS_DEMOTED = 3 MS_NORMAL = 1 = 2 MDS_NORMAL = 1 = 2 MDS_PRESSEI MDS_HOTCHECKE MDP_SEPERATOR = 2 MDS_NORMAL = 1 = 2 DNS_PRESSEI DNS_DISABLED = MDS_DISABLED = DNS_NORMAL = 1 = 2 DNS_PRESSEI DNS_DISABLED = DNHZS_NORMAL = 1 = 2 DNS_PRESSEI DNS_DISABLED = DNHZS_NORMAL = 2		MP_MENUBARITEM = 3	MS_SELECTED = :
MP_CHEVRON = 5 MP_CHEVRON = 5 MS_NORMAL = 1 MP_MENUDROPDOWN = 2 MS_DEMOTED = 3 MS_NORMAL = 1 MP_MENUITEM = 1 MP_MENUITEM = 1 MP_SEPARATOR = 6 MS_DEMOTED = 3 MS_NORMAL = 1 MS_SELECTED = : MS_DEMOTED = 3 MS_NORMAL = 1 MS_SELECTED = : MS_DEMOTED = 3 MS_NORMAL = 1 = 2 MDS_PRESSEI MDS_NORMAL = 1 = 2 MDS_PRESSEI MDS_CHECKED = MDS_HOTCHECKE MDP_SEPERATOR = 2 PAGE PGRP_DOWN = 2 PAGE PGRP_DOWN = 2 MS_NORMAL = 1 = 2 DNS_NORMAL = 1 = 2 DNS_NORMAL = 1 = 2 DNS_NORMAL = 1 = 2 DNS_NORMAL = 1 DNS_NORMAL = 1 MDS_DISABLED = DNHZS_NORMAL = 2 DNHZS_NORMAL = 2 DNHZS_			MS_DEMOTED = 3
MP_CHEVRON = 5 MS_SELECTED = : MS_DEMOTED = 3 MS_NORMAL = 1 MP_MENUDROPDOWN = 2 MS_SELECTED = : MS_DEMOTED = 3 MS_NORMAL = 1 MP_MENUITEM = 1 MP_SEPARATOR = 6 MS_NORMAL = 1 MS_SELECTED = : MS_DEMOTED = 3 MS_NORMAL = 1 = 2 MDS_NORMAL = 1 = 2 MDS_PRESSE MDS_NORMAL = 1 = 2 MDS_PRESSE MDS_HOTCHECKE MDP_SEPERATOR = 2 PAGE PGRP_DOWN = 2 PAGE PGRP_DOWN = 2 MS_NORMAL = 1 = 2 DNS_NORMAL = 1 = 2 DNS_PRESSE DNS_NORMAL = 1 = 2 DNS_PRESSE DNS_			MS_NORMAL = 1
MS_DEMOTED = 3 MS_NORMAL = 1 MP_MENUDROPDOWN = 2 MS_SELECTED = 3 MS_NORMAL = 1 MP_MENUITEM = 1 MP_SEPARATOR = 6 MS_NORMAL = 1 MS_SELECTED = 3 MS_NORMAL = 1 MS_SELECTED = 3 MS_NORMAL = 1 MS_DEMOTED = 3 MS_DEMOTED = 3 MDS_NORMAL = 1 = 2 MDS_PRESSEI MDS_NORMAL = 1 = 2 MDS_PRESSEI MDS_CHECKED = MDS_CHECKED = MDS_HOTCHECKE MDP_SEPERATOR = 2 MDS_NORMAL = 1 = 2 DNS_NORMAL = 1 D		$MP_CHEVRON = 5$	MS_SELECTED = :
MP_MENUDROPDOWN = 2 MP_MENUDROPDOWN = 2 MS_SELECTED = : MS_DEMOTED = 3 MS_NORMAL = 1 MP_MENUITEM = 1 MP_SEPARATOR = 6 MS_NORMAL = 1 MS_SELECTED = : MS_DEMOTED = 3 MS_NORMAL = 1 MS_SELECTED = : MS_DEMOTED = 3 MS_NORMAL = 1 = 2 MDS_NORMAL = 1 = 2 MDS_PRESSEI MDS_LISABLED = MDS_HOTCHECKED = MDS_HOTCHECKE MDP_SEPERATOR = 2 MDS_NORMAL = 1 = 2 DNS_NORMAL = 1 = 2 DNS_PRESSEI DNS_NORMAL = 1 = 2 DNS_PRESSEI DNS_DISABLED = DNHZS_NORMAL = 1 = 2 DNS_PRESSEI DNHZS_NORMAL = 1 = 2 DNS_PRESSEI = DNHZS_NORMAL = 1 = DNHZS_NORMAL = 1 = DNHZS_NORMAL = 1 = DNHZS_NORMAL			MS_DEMOTED = 3
MP_MENUDROPDOWN = 2 MS_SELECTED = , MS_DEMOTED = 3 MS_NORMAL = 1 MP_MENUITEM = 1 MP_SEPARATOR = 6 MS_NORMAL = 1 MS_SELECTED = ; MS_DEMOTED = 3 MS_NORMAL = 1 = 2 MDS_PRESSE MDS_NORMAL = 1 = 2 MDS_PRESSE MDS_LISABLED = MDS_HOTCHECKE MDP_SEPERATOR = 2 MDS_NORMAL = 1 = 2 DNS_NORMAL = 1 D			MS_NORMAL = 1
MP_MENUITEM = 1 MP_MENUITEM = 1 MP_SEPARATOR = 6 MS_NORMAL = 1 MS_DEMOTED = 3 MS_NORMAL = 1 MS_DEMOTED = 3 MS_DEMOTED = 3 MS_DEMOTED = 3 MS_DEMOTED = 3 MDS_NORMAL = 1 = 2 MDS_PRESSEI MDS_DISABLED = MDS_HOTCHECKE MDP_SEPERATOR = 2 PAGE PGRP_DOWN = 2 PAGE PGRP_DOWN = 2 PAGE PGRP_DOWN = 2 MS_NORMAL = 1 = 2 DNS_NORMAL = 1 = 2 DNS_PRESSEI DNS_NORMAL = 1 = 2 DNS_PRESSEI DNS_DISABLED = DNHZS_NORMAL = 1 = 2 DNS_PRESSEI DNHZS_NORMAL = 1 DNHZS_NORMAL = 1 DNHZS_NORMAL = 1 DNHZS_NORMAL = 1 DNHZS_NORMAL = 1 DNHZS_NORM		MP_MENUDROPDOWN = 2	MS_SELECTED = 7
MP_MENUITEM = 1 MP_MENUITEM = 1 MP_SEPARATOR = 6 MP_SEPARATOR = 6 MS_DEMOTED = 3 MS_NORMAL = 1 MS_DEMOTED = 3 MDS_NORMAL = 1 = 2 MDS_PRESSEI MDS_NORMAL = 1 = 2 MDS_PRESSEI MDS_CHECKED = MDS_CHECKED = MDS_HOTCHECKE MDP_SEPERATOR = 2 PAGE PGRP_DOWN = 2 PAGE PGRP_DOWN = 2 PAGE PGRP_DOWN = 2 MS_NORMAL = 1 = 2 DNS_NORMAL = 1 = 2 DNS_PRESSEI DNS_NORMAL = 1 = 2 DNS_PRESSEI DNS_NORMAL = 1 = 2 DNS_NORMAL = 1			MS MORMAL = 1
MB_DELOTIENT T MS_DEMOTED = 3 MS_NORMAL = 1 MS_SELECTED = 3 MS_DEMOTED = 3 MS_DEMOTED = 3 MDS_NORMAL = 1 = 2 MDS_PRESSE MDS_DISABLED = MDS_DISABLED = MDS_CHECKED = MDS_HOTCHECKE MDP_SEPERATOR = 2 PAGE PGRP_DOWN = 2 PAGE PGRP_DOWN = 2 DNS_NORMAL = 1 = 2 DNS_PRESSEI DNS_NORMAL = 1 = 2 DNS_PRESSEI = 2 DNS_PRESSEI		MP MENUITEM = 1	$MS_SELECTED = 1$
MP_SEPARATOR = 6 MP_SEPARATOR = 6 MS_NORMAL = 1 MS_DEMOTED = 3 MDS_NORMAL = 1 = 2 MDS_PRESSEI MDS_DISABLED = MDS_CHECKED = MDS_HOTCHECKE MDP_SEPERATOR = 2 PAGE PGRP_DOWN = 2 PAGE PGRP_DOWN = 2 DNS_NORMAL = 1 = 2 DNS_PRESSEI DNS_DISABLED = DNS_SEPERATOR = 2 DNS_NORMAL = 1 = 2 DNS_PRESSEI DNS_DISABLED = DNHZS_NORMAL = 1 = 2 DNS_PRESSEI DNS_NORMAL = 1 = 2 DNS_PRESSEI = 2 DNS			MS DEMOTED = 3
MP_SEPARATOR = 6 MS_SELECTED = : MS_DEMOTED = 3 MDS_NORMAL = 1 = 2 MDS_PRESSE MDS_DISABLED = MDS_CHECKED = MDS_HOTCHECKE MDP_SEPERATOR = 2 PAGE PGRP_DOWN = 2 PAGE PGRP_DOWN = 2 PGRP_DOWN = 2 MS_SELECTED = : MS_SELECTED = : MDS_NORMAL = 1 = 2 DNS_NORMAL = 1 = 2 DNS_PRESSEI DNS_DISABLED = DNHZS_NORMAL = 2 DNHZS_NORMAL = 2			MS NORMAL = 1
MENUBAND MDP_NEWAPPBUTTON = 1 MENUBAND MDP_NEWAPPBUTTON = 1 MDS_DISABLED = MDS_CHECKED = MDS_HOTCHECKE MDP_SEPERATOR = 2 PAGE PGRP_DOWN = 2 PAGE PGRP_DOWN = 2 MDS_NORMAL = 1 = 2 DNS_PRESSEI DNS_DISABLED = DNHZS_NORMAL = DNHZS_NORMAL = 2		$MP_SEPARATOR = 6$	MS_SELECTED = :
MENUBAND MDP_NEWAPPBUTTON = 1 MDP_NEWAPPBUTTON = 1 MDS_DISABLED = MDS_CHECKED = MDS_HOTCHECKE MDP_SEPERATOR = 2 PAGE PGRP_DOWN = 2 PAGE PGRP_DOWN = 2 DNS_NORMAL = 1 = 2 DNS_PRESSEI DNS_DISABLED = DNHZS_NORMAL = 2 DNHZS_NORMAL = 2			MS_DEMOTED = 3
MENUBAND MDP_NEWAPPBUTTON = 1 = 2 MDS_PRESSEI MDS_DISABLED = MDS_CHECKED = MDP_SEPERATOR = 2 MDS_HOTCHECKE PAGE PGRP_DOWN = 2 DNS_NORMAL = 1 = 2 DNS_PRESSEI DNS_DISABLED = DNS_DISABLED = DNHZS_NORMAL = 2			MDS_NORMAL = 1
MENUBAND MDP_NEWAPPBUTTON = 1 MDS_DISABLED = MDS_CHECKED = MDS_HOTCHECKE MDP_SEPERATOR = 2 PAGE PGRP_DOWN = 2 DNS_NORMAL = 1 = 2 DNS_PRESSEI DNS_DISABLED = DNHZS_NORMAL = 2			= 2 MDS_PRESSE
MDS_CHECKED = MDS_HOTCHECKE MDP_SEPERATOR = 2 PAGE PGRP_DOWN = 2 PAGE PGRP_DOWN = 2 DNS_NORMAL = 1 = 2 DNS_PRESSEI DNS_DISABLED = DNHZS_NORMAL = 2	MENUBAND	MDP_NEWAPPBUTTON = 1	MDS_DISABLED =
MDP_SEPERATOR = 2 PAGE PGRP_DOWN = 2 PAGE PGRP_DOWN = 2 DNS_NORMAL = 1 = 2 DNS_PRESSEL DNS_DISABLED = DNHZS_NORMAL = 2			MDS_CHECKED =
PAGE PGRP_DOWN = 2 DNS_NORMAL = 1 = 2 DNS_PRESSEL DNS_DISABLED = DNHZS_NORMAL = 2 DNHZS_HOT = 2			MDS_HOTCHECKE
PAGE PGRP_DOWN = 2 = 2 DNS_PRESSEI DNS_DISABLED = DNHZS_NORMAL = 1 DNHZS_HOT = 2		MDP_SEPERATOR = 2	
DNS_DISABLED = DNHZS_NORMAL = DNHZS_HOT = 2	PAGE	PCRP DOWN = 2	
DNHZS_NORMAL =	TAGE		DNS DISABLED =
DNHZS HOT = 2			DNH7S NORMAL =
			DNHZS HOT = 2
PGRP_DOWNHORZ = 4 DNHZS_PRESSED		$PGRP_DOWNHORZ = 4$	DNHZS_PRESSED

	PGRP_UP = 1	DNHZS_DISABLED UPS_NORMAL = 1 = 2 UPS_PRESSEL UPS_DISABLED =
	PGRP_UPHORZ = 3	UPHZS_NORMAL = UPHZS_HOT = 2 UPHZS_PRESSED UPHZS_DISABLED
PROGRESS	PP_BAR = 1	
	$PP_BARVERT = 2$ $PP_CHUNK = 3$	
	PP_CHUNKVERT = 4	
REBAR	RP_BAND = 3	
	RP_CHEVRON = 4	CHEVS_NORMAL = CHEVS_HOT = 2 CHEVS_PRESSED
	$RP_CHEVRONVERT = 5$ $RP_GRIPPER = 1$ $RP_GRIPPERVERT = 2$	_
SCROLLBAR	SBP_ARROWBTN = 1	ABS_DOWNDISAB ABS_DOWNHOT, ABS_DOWNNORM ABS_DOWNPRESS ABS_UPDISABLED ABS_UPHOT, ABS_UPHOT, ABS_UPNORMAL, ABS_UPPRESSED, ABS_LEFTDISABLI ABS_LEFTDISABLI ABS_LEFTHOT, ABS_LEFTPRESSE ABS_LEFTPRESSE ABS_RIGHTDISAB ABS_RIGHTHOT, ABS_RIGHTNORM ABS_RIGHTPRESS
	SBP_GRIPPERHORZ = 8	
	SDF_GRIFFERVERI = 9	SCRBS NORMAL :
	SBP_LOWERTRACKHORZ = 4	SCRBS_HOT = 2 SCRBS_PRESSED

	SBP_LOWERTRACKVERT = 6	SCRBS_DISABLEE SCRBS_NORMAL = SCRBS_HOT = 2 SCRBS_PRESSED SCRBS_DISABLEE
	SBP_THUMBBTNHORZ = 2	SCRBS_NORMAL = SCRBS_HOT = 2 SCRBS_PRESSED SCRBS_DISABLEE
	SBP_THUMBBTNVERT = 3	SCRBS_NORMAL = SCRBS_HOT = 2 SCRBS_PRESSED SCRBS_DISABLEE
	SBP_UPPERTRACKHORZ = 5	SCRBS_NORMAL = SCRBS_HOT = 2 SCRBS_PRESSED SCRBS_DISABLEE
	SBP_UPPERTRACKVERT = 7	SCRBS_NORMAL = SCRBS_HOT = 2 SCRBS_PRESSED SCRBS_DISABLEE
	SBP_SIZEBOX = 10	SZB_RIGHTALIGN SZB_LEFTALIGN =
SPIN	SPNP_DOWN = 2	DNS_NORMAL = 1 = 2 DNS_PRESSEI DNS_DISABLED =
	SPNP_DOWNHORZ = 4	DNHZS_NORMAL = DNHZS_HOT = 2 DNHZS_PRESSED
	SPNP_UP = 1	UPS_NORMAL = 1 = 2 UPS_PRESSEE UPS_DISABLED =
	SPNP_UPHORZ = 3	UPHZS_NORMAL = UPHZS_HOT = 2 UPHZS_PRESSED UPHZS_DISABLED
STARTPANEL	SPP LOGOFF = 8	
	_ SPP_LOGOFFBUTTONS = 9	SPLS_NORMAL = SPLS_HOT = 2 SPLS_PRESSED =

	SPP_MOREPROGRAMS = 2	
	SPP_MOREPROGRAMSARROW = 3	SPS_NORMAL = 1 = 2 SPS_PRESSEE
	SPP_PLACESLIST = 6	_
	SPP_PLACESLISTSEPARATOR = 7	
	SPP_PREVIEW = 11	
	SPP_PROGLIST = 4	
	SPP_PROGLISTSEPARATOR = 5	
	SPP_USERPANE = 1	
	SPP_USERPICTURE = 10	
STATUS	SP_GRIPPER = 3	
	SP_PANE = 1	
	SP_GRIPPERPANE = 2	
TAB	TABP_BODY = 10	
	TABP_PANE = 9	
	TABP_TABITEM = 1	TIS_NORMAL = 1 2 TIS_SELECTED = TIS_DISABLED = 4 TIS_FOCUSED = 5

TABP_TABITEMBOTHEDGE = 4

TABP_TABITEMLEFTEDGE = 2

TABP_TABITEMRIGHTEDGE = 3

TABP_TOPTABITEM = 5

TIBES NORMAL = TIBES HOT = 2TIBES_SELECTED TIBES DISABLED TIBES FOCUSED TILES NORMAL = TILES HOT = 2 TILES SELECTED TILES_DISABLED : TILES_FOCUSED = TIRES NORMAL = TIRES HOT = 2TIRES SELECTED TIRES DISABLED TIRES_FOCUSED TTIS NORMAL = 1 = 2 TTIS SELECTE TTIS DISABLED = TTIS_FOCUSED = TTIBES_NORMAL

	TABP_TOPTABITEMBOTHEDGE = 8 TABP_TOPTABITEMLEFTEDGE = 6	TTIBES_HOT = 2 TTIBES_SELECTE TTIBES_DISABLEE TTIBES_FOCUSEE TTILES_NORMAL = TTILES_HOT = 2 TTILES_SELECTEI TTILES_DISABLEE TTILES_FOCUSED
	TABP_TOPTABITEMRIGHTEDGE = 7	TTIRES_NORMAL TTIRES_HOT = 2 TTIRES_SELECTE TTIRES_DISABLEE TTIRES_FOCUSEE
TASKBAND	$TDP_GROUPCOUNT = 1$	
	$TDP_FLASHBUTTON = 2$	
	TDP_FLASHBUTTONGROUPMENU = 3	
TASKBAR	TBP_BACKGROUNDBOTTOM = 1	
	TBP_BACKGROUNDLEFT = 4	
	TBP_BACKGROUNDRIGHT = 2	
	$TBP_BACKGROUNDTOP = 3$	
	TBPSIZINGBARBOTTOM = 5	
	TBP_SIZINGBARBOTTOMLEFT = 8	
	TBP SIZINGBARRIGHT = 6	
	TBP SIZINGBARTOP = 7	
TOOLBAR	TP_BUTTON = 1	TS_NORMAL = 1 T TS_PRESSED = 3 TS_DISABLED = 4 TS_CHECKED = 5 TS_HOTCHECKED
	TP_DROPDOWNBUTTON = 2	TS_NORMAL = 1 T TS_PRESSED = 3 TS_DISABLED = 4 TS_CHECKED = 5 TS_HOTCHECKED TS_NORMAL = 1 T
	TP_SPLITBUTTON = 3	TS_PRESSED = 3 TS_DISABLED = 4 TS_CHECKED = 5

		TS_HOTCHECKED
		TS_NORMAL = 1 T
		TS_PRESSED = 3
	TP_SPLITBUTTONDROPDOWN = 4	TS_DISABLED = 4
		TS_CHECKED = 5
		TS_HOTCHECKED
		TS_NORMAL = 1 T
		TS_PRESSED = 3
	$TP_SEPARATOR = 5$	$TS_DISABLED = 4$
		TS_CHECKED = 5
		IS_HOTCHECKED
		TS_NORMAL = 1 T
		IS_PRESSED = 3
	IP_SEPARATORVERT = 6	IS_DISABLED = 4
		TS_UNTCHECKED = 5
TOOLTIP	$TTP_BALLOON = 3$	TTBS_NORMAL = 1
		TTBS_LINK - 2
	TTP_BALLOONTITLE = 4	TTBS_NORMAL =
		TIDS_LINK = 2
		TTCS_NORMAL =
	$TIP_CLOSE = 5$	$11CS_HO1 = 2$
		TTCO_PRESSED -
	$TTP_STANDARD = 1$	TISS_NORMAL =
		TTOO NORMAL
	TTP_STANDARDTITLE = 2	TISS_NORMAL =
	_	$1155_LINK = 2$
		TUS_NORMAL = 1
TRACKBAR	TKP_THUMB = 3	2 TUS_PRESSED =
		TUS_FOCUSED = 4
		TUBS_NORMAL -
		TUBS_PRESSED =
		TUBS_FOCUSED =
		TUBS DISABLED =
		TUVIS NORMAL =
		TUVLS HOT = 2
	TKP THUMBLEFT = 7	TUVLS PRESSED
		TUVLS_FOCUSED
		TUVLS DISABLED

	TKP_THUMBRIGHT = 8	TUVRS_NORMAL = TUVRS_HOT = 2 TUVRS_PRESSED TUVRS_FOCUSED TUVRS_DISABLED TUTS_NORMAL =
	TKP_THUMBTOP = 5	TUTS_PRESSED = TUTS_FOCUSED = TUTS_DISABLED =
	TKP_THUMBVERT = 6	TUVS_NORMAL = TUVS_HOT = 2 TUVS_PRESSED = TUVS_FOCUSED = TUVS_DISABLED =
	TKP_TICS = 9	TSS_NORMAL = 1
	TKP_TICSVERT = 10	TSVS_NORMAL =
	TKP_TRACK = 1	TRS_NORMAL = 1
	TKP_TRACKVERT = 2	TRVS_NORMAL =
TRAYNOTIFY	$TNP_ANIMBACKGROUND = 2$	
	TNP_BACKGROUND = 1	
TREEVIEW	TVP_BRANCH = 3	
	TVP_GLYPH = 2	GLPS_CLOSED = GLPS_OPENED =
	TVP_TREEITEM = 1	TREIS_NORMAL = TREIS_HOT = 2 TREIS_SELECTED TREIS_DISABLED TREIS_SELECTED = 5
WINDOW	WP CAPTION = 1	CS_ACTIVE = 1 CS
		= 2 CS_DISABLED
	$WP_CAPTIONSIZING TEMPLATE = 30$	CPS NOPMAL = 1
	WP_CLOSEBUTTON = 18	= 2 CBS_PUSHED CBS_DISABLED =
	WP_DIALOG = 29	
	WP_FRAMEBOTTOM = 9	FS_ACTIVE = 1 FS = 2
	WP FRAMEBOTTOMSIZINGTEMPLATE = 36	

WP_FRAMELEFT = 7
WP_FRAMELEFTSIZINGTEMPLATE = 32
WP_FRAMERIGHT = 8
WP_FRAMERIGHTSIZINGTEMPLATE = 34
WP_HELPBUTTON = 23
WP_HORZSCROLL = 25
WP_HORZTHUMB = 26
WP_MAX_BUTTON
WP_MAXCAPTION = 5
WP_MDICLOSEBUTTON = 20
WP_MDIHELPBUTTON = 24
WP_MDIMINBUTTON = 16
WP_MDIRESTOREBUTTON = 22
WP_MDISYSBUTTON = 14

WP MINBUTTON = 15

FS ACTIVE = 1 FS= 2 FS ACTIVE = 1 FS= 2 HBS NORMAL = 1= 2 HBS_PUSHED HBS DISABLED = HSS NORMAL = 1= 2 HSS PUSHED HSS_DISABLED = HTS NORMAL = 12 HTS PUSHED = 3 HTS_DISABLED = MAXBS NORMAL MAXBS HOT = 2MAXBS PUSHED = MAXBS_DISABLEE MXCS ACTIVE = 1 MXCS_INACTIVE = MXCS DISABLED CBS NORMAL = 1= 2 CBS PUSHED CBS_DISABLED = HBS NORMAL = 1 = 2 HBS_PUSHED HBS DISABLED = MINBS NORMAL = MINBS HOT = 2MINBS PUSHED = MINBS_DISABLED RBS NORMAL = 1 = 2 RBS_PUSHED RBS_DISABLED = SBS NORMAL = 1 = 2 SBS_PUSHED SBS_DISABLED = MINBS NORMAL = MINBS HOT = 2MINBS_PUSHED =

WP_MINCAPTION = 3	MINBS_DISABLED MNCS_ACTIVE = 1 MNCS_INACTIVE = MNCS_DISABLED
WP_RESTOREBUTTON = 21	RBS_NORMAL = 1 = 2 RBS_PUSHED RBS_DISABLED =
WP_SMALLCAPTION = 2	CS_ACTIVE = 1 CS = 2 CS_DISABLED
WP_SMALLCAPTIONSIZINGTEMPLATE = 31	
WP_SMALLCLOSEBUTTON = 19	CBS_NORMAL = 1 = 2 CBS_PUSHED CBS_DISABLED =
WP_SMALLFRAMEBOTTOM = 12	FS_ACTIVE = 1 FS = 2
WP_SMALLFRAMEBOTTOMSIZINGTEMPLATE = 37	
WP_SMALLFRAMELEFT = 10	FS_ACTIVE = 1 FS
WP_SMALLFRAMELEFTSIZINGTEMPLATE = 33	
WP_SMALLFRAMERIGHT = 11	FS_ACTIVE = 1 FS
WP_SMALLFRAMERIGHTSIZINGTEMPLATE = 35	- <i>L</i>
WP_SMALLHELPBUTTON	HBS_NORMAL = 1 = 2 HBS_PUSHED HBS_DISABLED =
WP_SMALLMAXBUTTON	MAXBS_NORMAL MAXBS_HOT = 2 MAXBS_PUSHED = MAXBS_DISABLEE
WP_SMALLMAXCAPTION = 6	MXCS_ACTIVE = 1 MXCS_INACTIVE = MXCS_DISABLED
WP_SMALLMINCAPTION = 4	MNCS_ACTIVE = 1 MNCS_INACTIVE = MNCS_DISABLED
WP_SMALLRESTOREBUTTON	RBS_NORMAL = 1 = 2 RBS_PUSHED

	RBS_DISABLED =
	SBS_NORMAL = 1
WP_SMALLSYSBUTTON	= 2 SBS_PUSHED
	SBS_DISABLED =
	SBS_NORMAL = 1
WP_SYSBUTTON = 13	= 2 SBS_PUSHED
	SBS_DISABLED =
	VSS_NORMAL = 1
WP_VERTSCROLL = 27	= 2 VSS_PUSHED
	VSS_DISABLED =
	VTS_NORMAL = 1
WP_VERTTHUMB = 28	2 VTS_PUSHED =
	VTS DISABLED =

method Appearance.Clear ()

Removes all skins in the control.

Туре

Description

Use the Clear method to clear all skins from the control. Use the <u>Remove</u> 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 control provides the <u>VisualDesign</u> property that allows you to easily change the control's visual appearance at design mode. Also, the VisualDesign property can be used at runtime to specify a visual appearance, by setting the VisualDesign property with a new generated value. The <u>UseVisualTheme</u> property indicates whether the current visual theme is applied to parts of the control.

method Appearance.Remove (ID as Long)

Removes a specific skin from the control.

Туре	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 <u>Add</u> method. Use the <u>Clear</u> 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 control provides the <u>VisualDesign</u> property that allows you to easily change the control's visual appearance at design mode. Also, the VisualDesign property can be used at runtime to specify a visual appearance, by setting the VisualDesign property with a new generated value. The <u>UseVisualTheme</u> property indicates whether the current visual theme is applied to parts of the control.

property Appearance.RenderType as Long

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

Туре	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 <u>EBNColor</u> tool to visualize applying EBN colors.

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

For instance, the following sample changes the events' appearance, by using an EBN object:

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

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

 "A" in Red (RGB(255,0,0), for instance the event's property BodyBackColor is 0x10000FF

- "B" in Green (RGB(0,255,0), for instance the event's property BodyBackColor is 0x100FF00
- "C" in Blue (RGB(0,0,255), for instance the event's property BodyBackColor is 0x1FF0000
- "Default", no color is specified, for instance the event's property BodyBackColor is 0x1000000

The RenderType property could be one of the following:

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

	Wednesday, June 20, 2012	
08:00 AM		
:30	8:30 AM - 10:30 AM	
MA 00:00		
:30		
10:00 AM	A	
:30	10:30 AM - 12:30 PM	
11:00 AM		
:30		
12:00 PM	в	
:30	12:30 PM - 2:30 PM	
01:00 PM		
:30		
02:00 PM	с	2:00 PM - 4:00 PM
:30		
03:00 PM		
:30		Default

-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 BodyEventBackColor = &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(0,255,0), RGB(0,255,0,0), RGB(127,0,0), RGB(0,127,0), ...)

	Wednesday, June 20, 2012	
08:00 AM		
:30	8:30 AM - 10:30 AM	
MA 00:00		
:30		
10:00 AM	A	
:30	10:30 AM - 12:30 PM	
11:00 AM		
:30		
12:00 PM	В	
:30	12:30 PM - 2:30 PM	
01:00 PM		
:30		
02:00 PM	С	2:00 PM - 4:00 PM
:30		
03:00 PM		
:30		Default

-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 BodyEventBackColor = &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(0,255,0), RGB(0,255,0), RGB(0,255,0), ...)

	Wednesday, June 20, 2012	
08:00 AM		
:30	8:30 AM - 10:30 AM	
MA 00:00		
:30		
10:00 AM	A	
:30	10:30 AM - 12:30 PM	
11:00 AM		
:30		
12:00 PM	В	
:30	12:30 PM - 2:30 PM	
01:00 PM		
:30		
02:00 PM	C .	2:00 PM - 4:00 PM
:30		
03:00 PM		
:30		Default

• *0, default*, the specified color is applied to the EBN. For instance, the BodyEventBackColor = &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.

	Wednesday, June 20, 2012	
08:00 AM		
:30	8:30 AM - 10:30 AM	
09:00 AM		
:30		
10:00 AM	A	
:30	10:30 AM - 12:30 PM	
11:00 AM		
:30		
12:00 PM	в	
:30	12:30 PM - 2:30 PM	
01:00 PM		
:30		
02:00 PM	C 2:00 PM	1 - 4:00 PM
:30		
03:00 PM		
:30	Default	

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):

	Wednesday, June 20, 2012	
08:00 AM		
:30	8:30 AM - 10:30 AM	
MA 00:00		
:30		
10:00 AM	A	
:30	10:30 AM - 12:30 PM	
11:00 AM		
:30		
12:00 PM	в	
:30	12:30 PM - 2:30 PM	
01:00 PM		
:30		
02:00 PM	с	2:00 PM - 4:00 PM
:30		
03:00 PM		
:30		Default

The following picture shows the control with the RenderType property on 0x8000FFFF (50% Yellow, 0x80 or 128 in decimal is 50% from 256):

	Wednesday, June 20, 2012	
08:00 AM		
:30	8:30 AM - 10:30 AM	
09:00 AM		
:30		
10:00 AM	A	
:30	10:30 AM - 12:30 PM	
11:00 AM		
:30		
12:00 PM	В	
:30	12:30 PM - 2:30 PM	
01:00 PM		
:30		
02:00 PM	с	2:00 PM - 4:00 PM
:30		
03:00 PM		
:30		Default

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):

	Wednesday, June 20, 2012	
08:00 AM		
:30	8:30 AM - 10:30 AM	
09:00 AM		
:30		
10:00 AM	A	
:30	10:30 AM - 12:30 PM	
11:00 AM		
:30		
12:00 PM	в	
:30	12:30 PM - 2:30 PM	
01:00 PM		
:30		
02:00 PM	С	2:00 PM - 4:00 PM
:30		
03:00 PM		
:30		Default

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):



Calendar object

The Calendar object indicates the calendar panel of the component like in the following screen shot:



You can access the component's Calendar object using the <u>Calendar</u> property. Use the <u>OnResizeControl</u> property to specify one of the followings:

- auto hide the calendar panel. Ability to hide the calendar section while the cursor is not in it (OnResizeControlEnum.exResizePanelRight Or OnResizeControlEnum.exCalendarFit Or OnResizeControlEnum.exCalendarAutoHide).
- hide completely the calendar section (exHideSplitter)
- specify the alignment of the calendar, as on the left or right side of the schedule view (OnResizeControlEnum.exChangePanels Or OnResizeControlEnum.exCalendarFit)
- full or partially view of the calendar panel (exResizePanelRight)

Name	Description
<u>AlignDate</u>	Specifies whether the January 1st or MinDate is aligned to the top-left corner.
<u>AllowFocusDate</u>	Specifies the combination of keys that allows the user to focus a new date, in the calendar panel.
<u>AllowSelectDate</u>	Specifies the combination of keys that allows the user to select dates in the calendar panel.
<u>AllowSelectDateRect</u>	Specifies the combination of keys that allows the user to select dates in the calendar panel, by dragging a rectangle.

<u>AllowToggleSelectKey</u>	Specifies the combination of keys to select multiple not- contiguously dates.
AMPM	Specifies the AM and PM indicators.
Date	Retrieves or sets the date being displayed in the calendar panel.
DateFromPoint	Retrieves the date from the cursor, in the calendar panel.
DisableZoneFormat	Returns or sets an expression that determines the dates being disabled in the calendar/schedule panel.
DisplayWeekNumberAs	Specifies the way the calendar displays the week number.
<u>Events</u>	Returns a safe array of dates with events, in a specified group, where the Expression indicates the formula to determine the dates being verified.
<u>FirstVisibleDate</u>	Retrieves the first visible date, in the calendar panel.
<u>FirstWeekDay</u>	Specifies the first day of the week.
<u>FitSelToView</u>	Specifies the list of additional dates to be shown on the schedule view, when OnSelectDate property is exFitSelToView.
<u>FocusDate</u>	Retrieves the date being focused in the calendar panel.
<u>GroupHighlightEvent</u>	Highlights the date in the calendar panel using the HighlightEvent property of each Group found on day's events.
<u>HeaderDayLabel</u>	Specifies the HTML date-format to be shown on the calendar's header.
<u>HideSel</u>	Specifies whether selected date appears selected when a control loses focus.
<u>HighlightEvent</u>	Gives access to the Highlight object, so you can customize highlighting the events, in the calendar panel.
<u>hWnd</u>	Retrieves the calendar's window handle.
LastVisibleDate	Retrieves the last visible date, in the calendar panel.
LocAMPM	Retrieves the time marker such as AM or PM using the current user regional and language settings.
<u>LocFirstWeekDay</u>	Indicates the first day of the week, as specified in the regional settings.
LocMonthNames	Retrieves the list of month names, as indicated in the regional settings, separated by space.
	Retrieves the list of names for each week day, as

<u>LocWeekDays</u>	indicated in the regional settings, separated by space.
LongDateFormat	Indicates the long date format.
LongTimeFormat	Indicates the long time format.
<u>MaxDate</u>	Retrieves or sets the max date.
<u>MaxMonthX</u>	Specifies the maximum number of months horizontally displayed.
<u>MaxMonthY</u>	Specifies the maximum number of months vertically displayed.
<u>MinDate</u>	Retrieves or sets the min date.
<u>MinMonthX</u>	Specifies the minimum number of months horizontally displayed.
<u>MinMonthY</u>	Specifies the minimum number of months vertically displayed.
MonthNames	Retrieves or sets a value that indicates the list of month names, separated by space.
<u>NonworkingDays</u>	Retrieves or sets a value that indicates the non-working days, for each week day a bit.
<u>NonworkingDaysColor</u>	Retrieves or sets a value that indicates the color to fill the non-working days.
<u>NonworkingDaysFrameColor</u>	Retrieves or sets a value that indicates the color to show the non-working frame.
<u>NonworkingDaysPattern</u>	Retrieves or sets a value that indicates the pattern being used to fill non-working days.
<u>OnSelectDate</u>	Specifies the action that the control does once the user selects new dates in the calendar panel.
Parent	Specifies the handle of the window that hosts the calendar panel.
<u>SelCount</u>	Indicates the number of dates being selected in the calendar panel.
SelDate	Gets the date being selected giving its index in the selection.
<u>Select</u>	Gets the date being selected giving its index in the selection. Selects the current (focus) day, week, week day, month or year in the calendar panel.
<u>Select</u> <u>SelectDate</u>	Gets the date being selected giving its index in the selection.Selects the current (focus) day, week, week day, month or year in the calendar panel.Selects or unselects a date in the calendar panel.

<u>Selection</u>	calendar panel.
ShortDateFormat	Indicates the short date format.
ShortTimeFormat	Indicates the short time format.
ShowGridLines	Shows or hides the grid lines in the calendar panel.
<u>ShowHighlightEvent</u>	Returns or sets a value that indicates whether the calendar panel highlights days that contain events.
<u>ShowNonMonthDays</u>	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, in the calendar panel.
<u>ShowWeeks</u>	Retrieves or sets a value that indicates whether the weeks header is visible or hidden.
<u>ShowYearScroll</u>	Retrieves or sets a value that indicates whether the scroll bar (in the calendar panel) to change the year is visible or hidden.
<u>SingleSel</u>	Returns or sets a value that indicates whether the user can select one or more dates in the calendar panel.
TodayCaption	Retrieves or sets a value that indicates the today button's caption, in the calendar panel.
<u>WeekDays</u>	Retrieves or sets a value that indicates the list of names for each week day, separated by space.

property Calendar.AlignDate as Boolean

Specifies whether the January 1st or MinDate is aligned to the top-left corner.

Туре	Description
Boolean	A Boolean expression that specifies whether January 1st or MinDate is aligned to the top-left corner.

By default, the AlignDate property is True. This property has effect only if multiple months are displayed in the calendar panel. Use the <u>MinMonthX/MaxMonthX</u> and <u>MinMonthY/MaxMonthY</u> properties to specify the number of months to be displayed on the calendar panel.
property Calendar.AllowFocusDate as AllowKeysEnum

Specifies the combination of keys that allows the user to focus a new date, in the calendar panel.

Туре	Description
<u>AllowKeysEnum</u>	An AllowKeysEnum expression that specifies the combination of keys that allows the user to focus a new date, in the calendar panel.

By default, the AllowFocusDate property is exRightClick. Changing the focus using this option may show a different month, without *selecting* the focusing date. The control may display only a single focused date, but it can display multiple selection dates. The <u>FocusDate</u> property indicates the date being focused. The control fires the <u>LayoutStartChanging(exCalendarFocusDateChange)</u> when the user is about to change the focusing date, and the <u>LayoutEndChanging(exCalendarFocusDateChange)</u> notifies your application once a new date is being focused. The <u>Background(exCalendarFocusDateForeColor)</u> changes the visual appearance of the focused date, while the <u>Background(exCalendarFocusDateForeColor)</u> changes the foreground color of the focused date.

The AllowFocusDate property on exDisallow disables focusing a date that's not being selected. The <u>AllowSelectDate</u> property indicates the keys combination so the user can select new dates in the calendar panel, and so, new dates to be shown in the schedule view. The <u>Date</u> property of the control browses a new month in the calendar panel.

Use the <u>Selection/SelectDate</u> property to change programmatically the dates being selected in the calendar, including the dates to be shown in the schedule view. You can use the <u>Selection/SelCount/SelDate</u> property to retrieve the selected dates.

property Calendar.AllowSelectDate as AllowKeysEnum

Specifies the combination of keys that allows the user to select dates in the calendar panel.

Туре	Description
<u>AllowKeysEnum</u>	An AllowKeysEnum expression that specifies the combination of keys that allows the user to select new dates, in the calendar panel, and so to display a different dates in the schedule view.

By default, the AllowSelectDate property is exLeftClick. The exDisallow indicates the the user can not select dates in the calendar panel. Once the user starts selecting a new date in the calendar panel, the control fires the

LayoutStartChanging(exCalendarSelectionChange). Once a new date is selected, the LayoutEndChanging(exCalendarSelectionChange) event occurs. The <u>SingleSel</u> property indicates whether the user can select one or multiple dates. The <u>AllowSelectDateRect</u> specifies the keys combination so the user can do a rectangular selection in the calendar panel. Use the <u>Selection/SelectDate</u> property to change programmatically the dates being selected in the calendar, including the dates to be shown in the schedule view. You can use the <u>Selection/SelCount/SelDate</u> property to retrieve the selected dates.

- The <u>Background</u>(exCalendarSelBackColor), <u>Background</u>(exCalendarSelForeColor) property specifies the visual appearance of the selected dates, when the control has the focus.
- The <u>Background</u>(exCalendarSelBackColorUnFocus), <u>Background</u>(exCalendarSelBackColorUnFocus) property specifies the visual appearance of the selected dates, when the control has no focus.

The <u>ClipToSel</u> property indicates whether the control clips the schedule panel to view the selected dates only. The <u>FitSelToView</u> method restores the view to fit the selected dates.

property Calendar.AllowSelectDateRect as AllowKeysEnum

Specifies the combination of keys that allows the user to select dates in the calendar panel, by dragging a rectangle.

Туре	Description
<u>AllowKeysEnum</u>	An AllowKeysEnum expression that specifies the combination of keys that allows the user to rectangular select new dates, in the calendar panel, and so to display a different dates in the schedule view.

By default, the AllowSelectDateRect property is exLeftClick + exALTKey, which means that the user can hold the ALT key while pressing the left mouse button and so a rectangle is shown, and each date that intersect the rectangular region will be selected by dragging the mouse.

The exDisallow indicates the the user can not rectangular select dates in the calendar panel. Once the user starts selecting a new date in the calendar panel, the control fires the LayoutStartChanging(exCalendarSelectionChange). Once a new date is selected, the LayoutEndChanging(exCalendarSelectionChange) event occurs. The <u>SingleSel</u> property indicates whether the user can select one or multiple dates. The <u>AllowSelectDate</u> specifies the keys combination so the user can do a selection in the calendar panel. Use the <u>Selection/SelectDate</u> property to change programmatically the dates being selected in the calendar, including the dates to be shown in the schedule view. You can use the <u>Selection/SelCount/SelDate</u> property to retrieve the selected dates. You can do the same type of the selection in the schedule panel, by using the <u>AllowSelectEventRect</u> property.

The <u>ClipToSel</u> property indicates whether the control clips the schedule panel to view the selected dates only. The <u>FitSelToView</u> method restores the view to fit the selected dates.

The following screen shot shows the rectangular selection, in the calendar panel:

1			lune	201	2		•
	Su	Мо	Tu	We	Th	Fr	Sa
22						1	2
23	3	4	0	0	1	8	9
24	10	11	12	13	(4	15	16
25	17	18	19	20	1	22	23
26	24	25	26	27	\$	29	30
27					•	'	

property Calendar.AllowToggleSelectKey as AllowKeysEnum

Specifies the combination of keys to select multiple not-contiguously dates.

Туре	Description
<u>AllowKeysEnum</u>	An AllowKeysEnum expression that specifies the combination of keys to select multiple not-contiguously dates.

By default, the AllowToggleSelectKey property is exCTRLKey. This indicates that the user toggles the selected dates to unselected, and reverse by clicking the CTRL key. This property should be used in combination with <u>AllowSelectDate</u> and <u>AllowSelectDateRect</u> properties. Once the user starts selecting a new date in the calendar panel, the control fires the <u>LayoutStartChanging</u>(exCalendarSelectionChange). Once a new date is selected, the <u>LayoutEndChanging</u>(exCalendarSelectionChange) event occurs. Use the <u>SelectDate</u> property to change programmatically the dates being selected in the calendar, including the dates to be shown in the schedule view. You can use the <u>Selection/SelCount/SelDate</u> property to retrieve the selected dates.

The samples shows how you can toggle the selection of dates, such the first click selects the date, the second click unselects it, and so on.

VBA (MS Access, Excell...)

```
With Schedule1

.OnResizeControl = 1281

With .Calendar

.AllowToggleSelectKey = 1

.AllowSelectDate = 1

.SingleSel = False

End With

End With
```

VB6

```
With Schedule1
.OnResizeControl = OnResizeControlEnum.exResizePanelRight Or
OnResizeControlEnum.exHideSplitter Or OnResizeControlEnum.exCalendarFit
With .Calendar
.AllowToggleSelectKey = exLeftClick
.AllowSelectDate = exLeftClick
```

```
.SingleSel = False
End With
End With
```

VB.NET

```
With Exschedule1

.OnResizeControl =

exontrol.EXSCHEDULELib.OnResizeControlEnum.exResizePanelRight Or

exontrol.EXSCHEDULELib.OnResizeControlEnum.exHideSplitter Or

exontrol.EXSCHEDULELib.OnResizeControlEnum.exCalendarFit

With .Calendar

.AllowToggleSelectKey = exontrol.EXSCHEDULELib.AllowKeysEnum.exLeftClick

.AllowSelectDate = exontrol.EXSCHEDULELib.AllowKeysEnum.exLeftClick

.SingleSel = False

End With

End With
```

VB.NET for /COM

```
With AxSchedule1
.OnResizeControl = EXSCHEDULELib.OnResizeControlEnum.exResizePanelRight Or
EXSCHEDULELib.OnResizeControlEnum.exHideSplitter Or
EXSCHEDULELib.OnResizeControlEnum.exCalendarFit
With .Calendar
.AllowToggleSelectKey = EXSCHEDULELib.AllowKeysEnum.exLeftClick
.AllowSelectDate = EXSCHEDULELib.AllowKeysEnum.exLeftClick
.SingleSel = False
End With
End With
```

```
C++
```

Copy and paste the following directives to your header file as it defines the namespace 'EXSCHEDULELib' for the library: 'ExSchedule 1.0 Control Library'

```
#import <ExSchedule.dll>
using namespace EXSCHEDULELib;
*/
EXSCHEDULELib::ISchedulePtr spSchedule1 = GetDlgItem(IDC_SCHEDULE1)-
>GetControlUnknown();
spSchedule1-
>PutOnResizeControl(EXSCHEDULELib::OnResizeControlEnum(EXSCHEDULELib::exResiz
|EXSCHEDULELib::exHideSplitter | EXSCHEDULELib::exCalendarFit));
EXSCHEDULELib::ICalendarPtr var_Calendar = spSchedule1->GetCalendar();
var_Calendar->PutAllowToggleSelectKey(EXSCHEDULELib::exLeftClick);
var_Calendar->PutSingleSel(VARIANT_FALSE);
```

C++ Builder

Schedule1->OnResizeControl = Exschedulelib_tlb::OnResizeControlEnum::exResizePanelRight | Exschedulelib_tlb::OnResizeControlEnum::exHideSplitter | Exschedulelib_tlb::OnResizeControlEnum::exCalendarFit; Exschedulelib_tlb::ICalendarPtr var_Calendar = Schedule1->Calendar; var_Calendar->**AllowToggleSelectKey** = Exschedulelib_tlb::AllowKeysEnum::exLeftClick; var_Calendar->**AllowSelectDate** = Exschedulelib_tlb::AllowKeysEnum::exLeftClick; var_Calendar->SingleSel = false;

C#

exschedule1.OnResizeControl = exontrol.EXSCHEDULELib.OnResizeControlEnum.exResizePanelRight | exontrol.EXSCHEDULELib.OnResizeControlEnum.exHideSplitter | exontrol.EXSCHEDULELib.OnResizeControlEnum.exCalendarFit; exontrol.EXSCHEDULELib.Calendar var_Calendar = exschedule1.Calendar; var_Calendar.**AllowToggleSelectKey** = exontrol.EXSCHEDULELib.AllowKeysEnum.exLeftClick; var_Calendar.**AllowSelectDate** = exontrol.EXSCHEDULELib.AllowKeysEnum.exLeftClick;

```
var_Calendar.SingleSel = false;
```

JavaScript

```
<OBJECT classid="clsid:9B09E13D-7A88-4299-9DBE-383380435377"
id="Schedule1"></OBJECT>
```

```
<SCRIPT LANGUAGE="JScript">
Schedule1.OnResizeControl = 1281;
var var_Calendar = Schedule1.Calendar;
var_Calendar.AllowToggleSelectKey = 1;
var_Calendar.AllowSelectDate = 1;
var_Calendar.SingleSel = false;
</SCRIPT>
```

C# for /COM

```
axSchedule1.OnResizeControl =
EXSCHEDULELib.OnResizeControlEnum.exResizePanelRight |
EXSCHEDULELib.OnResizeControlEnum.exHideSplitter |
EXSCHEDULELib.OnResizeControlEnum.exCalendarFit;
EXSCHEDULELib.Calendar var_Calendar = axSchedule1.Calendar;
var_Calendar.AllowToggleSelectKey =
EXSCHEDULELib.AllowKeysEnum.exLeftClick;
var_Calendar.AllowSelectDate = EXSCHEDULELib.AllowKeysEnum.exLeftClick;
var_Calendar.SingleSel = false;
```

X++ (Dynamics Ax 2009)

```
public void init()
{
    COM com_Calendar;
    anytype var_Calendar;
    ;
    super();
```

```
exschedule1.OnResizeControl(1281/*exResizePanelRight | exHideSplitter |
exCalendarFit*/);
var_Calendar = exschedule1.Calendar(); com_Calendar = var_Calendar;
    com_Calendar.AllowToggleSelectKey(1/*exLeftClick*/);
    com_Calendar.AllowSelectDate(1/*exLeftClick*/);
    com_Calendar.SingleSel(false);
}
```

Delphi 8 (.NET only)

```
with AxSchedule1 do
begin
    OnResizeControl :=
Integer(EXSCHEDULELib.OnResizeControlEnum.exResizePanelRight) Or
Integer(EXSCHEDULELib.OnResizeControlEnum.exHideSplitter) Or
Integer(EXSCHEDULELib.OnResizeControlEnum.exCalendarFit);
    with Calendar do
    begin
    AllowToggleSelectKey := EXSCHEDULELib.AllowKeysEnum.exLeftClick;
    AllowSelectDate := EXSCHEDULELib.AllowKeysEnum.exLeftClick;
    SingleSel := False;
    end;
```

end

Delphi (standard)

```
with Schedule1 do
begin
OnResizeControl := Integer(EXSCHEDULELib_TLB.exResizePanelRight) Or
Integer(EXSCHEDULELib_TLB.exHideSplitter) Or
Integer(EXSCHEDULELib_TLB.exCalendarFit);
with Calendar do
begin
AllowToggleSelectKey := EXSCHEDULELib_TLB.exLeftClick;
AllowSelectDate := EXSCHEDULELib_TLB.exLeftClick;
SingleSel := False;
end;
```

end

VFP

with thisform.Schedule1
.OnResizeControl = 1281
with .Calendar
.AllowToggleSelectKey = 1
.AllowSelectDate = 1
.SingleSel = .F.
endwith
endwith

dBASE Plus

local oSchedule,var_Calendar oSchedule = form.Activex1.nativeObject oSchedule.OnResizeControl = 1281 /*exResizePanelRight | exHideSplitter | exCalendarFit*/ var_Calendar = oSchedule.Calendar var_Calendar.AllowToggleSelectKey = 1 var_Calendar.AllowSelectDate = 1 var_Calendar.SingleSel = false

XBasic (Alpha Five)

```
Dim oSchedule as P
Dim var_Calendar as P
oSchedule = topparent:CONTROL_ACTIVEX1.activex
oSchedule.OnResizeControl = 1281 'exResizePanelRight + exHideSplitter +
exCalendarFit
var_Calendar = oSchedule.Calendar
var_Calendar.AllowToggleSelectKey = 1
var_Calendar.AllowSelectDate = 1
var_Calendar.SingleSel = .f.
```

Visual Objects

local var_Calendar as ICalendar

```
oDCOCX_Exontrol1:OnResizeControl := exResizePanelRight | exHideSplitter |
exCalendarFit
var_Calendar := oDCOCX_Exontrol1:Calendar
var_Calendar:AllowToggleSelectKey := exLeftClick
var_Calendar:AllowSelectDate := exLeftClick
var_Calendar:SingleSel := false
```

PowerBuilder

```
OleObject oSchedule,var_Calendar

oSchedule = ole_1.Object

oSchedule.OnResizeControl = 1281 /*exResizePanelRight | exHideSplitter |

exCalendarFit*/

var_Calendar = oSchedule.Calendar

var_Calendar.AllowToggleSelectKey = 1

var_Calendar.AllowSelectDate = 1

var_Calendar.SingleSel = false
```

property Calendar.AMPM as String

Specifies the AM and PM indicators.

Туре	Description
String	A String expression that indicates the AM, PM time indicators to be shown in the control, separated by space.

By default, the AMPM property is "AM PM". Use the LocAMPM property to get the locale AM/PM indicators as indicated by current regional settings. The <%AM/PM%> HTML tag indicates the twelve-hour clock with the uppercase letters "AM" or "PM", as appropriate set by the AMPM property. The <%loc_AM/PM%> HTML tag indicates the time marker such as AM or PM using the current user regional and language settings (LocAMPM property). The FirstWeekDay property indicates the first day of the week. The MonthNames property specifies the list of name of the months. The WeekDays property specifies the name of the days in the week.

The following samples set the current view to display the locate date/time as set in the current regional settings.

VBA (MS Access, Excell...)

With Schedule1 With .Calendar .FirstWeekDay = .**LocFirstWeekDay** .MonthNames = .**LocMonthNames** .WeekDays = .**LocWeekDays** .AMPM = .**LocAMPM** End With End With

VB6

With Schedule1 With .Calendar .FirstWeekDay = .**LocFirstWeekDay** .MonthNames = .**LocMonthNames** .WeekDays = .**LocWeekDays** .AMPM = .**LocAMPM** End With End With With Exschedule1 With .Calendar .FirstWeekDay = .**LocFirstWeekDay** .MonthNames = .**LocMonthNames** .WeekDays = .**LocWeekDays** .AMPM = .**LocAMPM** End With End With

VB.NET for /COM

With AxSchedule1 With .Calendar .FirstWeekDay = .**LocFirstWeekDay** .MonthNames = .**LocMonthNames** .WeekDays = .**LocWeekDays** .AMPM = .**LocAMPM** End With End With

C++

```
/*
Copy and paste the following directives to your header file as
it defines the namespace 'EXSCHEDULELib' for the library: 'ExSchedule 1.0 Control
Library'
#import <ExSchedule.dll>
using namespace EXSCHEDULELib;
*/
EXSCHEDULELib::ISchedulePtr spSchedule1 = GetDlgItem(IDC_SCHEDULE1)-
>GetControlUnknown();
EXSCHEDULELib::ICalendarPtr var_Calendar = spSchedule1->GetCalendar();
var_Calendar->PutFirstWeekDay(var_Calendar->GetLocFirstWeekDay());
var_Calendar->PutWeekDays(var_Calendar->GetLocWeekDays());
```

```
var_Calendar->PutAMPM(var_Calendar->GetLocAMPM());
```

C++ Builder

Exschedulelib_tlb::ICalendarPtr var_Calendar = Schedule1->Calendar; var_Calendar->FirstWeekDay = var_Calendar->LocFirstWeekDay; var_Calendar->MonthNames = var_Calendar->LocMonthNames; var_Calendar->WeekDays = var_Calendar->LocWeekDays; var_Calendar->AMPM = var_Calendar->LocAMPM;

C#

exontrol.EXSCHEDULELib.Calendar var_Calendar = exschedule1.Calendar; var_Calendar.FirstWeekDay = var_Calendar.LocFirstWeekDay; var_Calendar.MonthNames = var_Calendar.LocMonthNames; var_Calendar.WeekDays = var_Calendar.LocWeekDays; var_Calendar.AMPM = var_Calendar.LocAMPM;

JavaScript

```
<OBJECT classid="clsid:9B09E13D-7A88-4299-9DBE-383380435377"
id="Schedule1"> </OBJECT>
</SCRIPT LANGUAGE="JScript">
var var_Calendar = Schedule1.Calendar;
var_Calendar.FirstWeekDay = var_Calendar.LocFirstWeekDay;
var_Calendar.MonthNames = var_Calendar.LocMonthNames;
var_Calendar.WeekDays = var_Calendar.LocWeekDays;
var_Calendar.AMPM = var_Calendar.LocAMPM;
</SCRIPT>
```

C# for /COM

EXSCHEDULELib.Calendar var_Calendar = axSchedule1.Calendar; var_Calendar.FirstWeekDay = var_Calendar.LocFirstWeekDay; var_Calendar.MonthNames = var_Calendar.LocMonthNames;

```
var_Calendar.WeekDays = var_Calendar.LocWeekDays;
var_Calendar.AMPM = var_Calendar.LocAMPM;
```

X++ (Dynamics Ax 2009)

```
public void init()
{
    COM com_Calendar;
    anytype var_Calendar;
    ;
    super();
    var_Calendar = exschedule1.Calendar(); com_Calendar = var_Calendar;
        com_Calendar.FirstWeekDay(com_Calendar.LocFirstWeekDay());
        com_Calendar.MonthNames(com_Calendar.LocMonthNames());
        com_Calendar.WeekDays(com_Calendar.LocWeekDays());
        com_Calendar.AMPM(com_Calendar.LocMPM());
    }
}
```

Delphi 8 (.NET only)

```
with AxSchedule1 do
begin
with Calendar do
begin
FirstWeekDay := LocFirstWeekDay;
MonthNames := LocMonthNames;
WeekDays := LocWeekDays;
AMPM := LocAMPM;
end;
end
```

Delphi (standard)

```
with Schedule1 do begin
```

```
with Calendar do
begin
    FirstWeekDay := LocFirstWeekDay;
    MonthNames := LocMonthNames;
    WeekDays := LocWeekDays;
    AMPM := LocAMPM;
end;
end
```

VFP

```
with thisform.Schedule1
with .Calendar
.FirstWeekDay = .LocFirstWeekDay
.MonthNames = .LocMonthNames
.WeekDays = .LocWeekDays
.AMPM = .LocAMPM
endwith
endwith
```

dBASE Plus

local oSchedule,var_Calendar

```
oSchedule = form.Activex1.nativeObject
var_Calendar = oSchedule.Calendar
var_Calendar.FirstWeekDay = var_Calendar.LocFirstWeekDay
var_Calendar.MonthNames = var_Calendar.LocMonthNames
var_Calendar.WeekDays = var_Calendar.LocWeekDays
var_Calendar.AMPM = var_Calendar.LocAMPM
```

XBasic (Alpha Five)

Dim oSchedule as P Dim var_Calendar as P

oSchedule = topparent:CONTROL_ACTIVEX1.activex

```
var_Calendar = oSchedule.Calendar
var_Calendar.FirstWeekDay = var_Calendar.LocFirstWeekDay
var_Calendar.MonthNames = var_Calendar.LocMonthNames
var_Calendar.WeekDays = var_Calendar.LocWeekDays
var_Calendar.AMPM = var_Calendar.LocAMPM
```

Visual Objects

local var_Calendar as ICalendar

```
var_Calendar := oDCOCX_Exontrol1:Calendar
var_Calendar:FirstWeekDay := var_Calendar:LocFirstWeekDay
var_Calendar:MonthNames := var_Calendar:LocMonthNames
var_Calendar:WeekDays := var_Calendar:LocWeekDays
var_Calendar:AMPM := var_Calendar:LocAMPM
```

PowerBuilder

```
OleObject oSchedule,var_Calendar
oSchedule = ole_1.Object
var_Calendar = oSchedule.Calendar
var_Calendar.FirstWeekDay = var_Calendar.LocFirstWeekDay
var_Calendar.MonthNames = var_Calendar.LocMonthNames
var_Calendar.WeekDays = var_Calendar.LocWeekDays
var_Calendar.MPM = var_Calendar.LocAMPM
```

property Calendar.Date as Date

l

Retrieves or sets the date being displayed in the calendar panel.

Туре	Description
Date	A DATE expression that indicates the date (month) being shown in the calendar panel.

The Date property indicates the date (month) being shown in the calendar panel. The <u>AllowSelectDate</u> property indicates the keys combination so the user can select new dates in the calendar panel, and so, new dates to be shown in the schedule view. Use the <u>Selection/SelectDate</u> property to change programmatically the dates being selected in the calendar, including the dates to be shown in the schedule view. You can use the <u>Selection/SelCount/SelDate</u> property to retrieve the selected dates. The <u>DateFromPoint</u> property indicates the date from the cursor in the calendar panel. The <u>DateTimeFromPoint</u> property indicates the date/time from the cursor on the schedule panel. The <u>TimeFromPoint</u> property indicates the time from the cursor on the schedule panel. The <u>TimeFromPoint</u> property indicates the time from the cursor on the schedule panel. The <u>TimeFromPoint</u> property indicates the time from the cursor on the schedule panel. The <u>TimeFromPoint</u> property indicates the time from the cursor on the schedule panel.

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

Retrieves the date from the cursor, in the calendar panel.

Туре	Description
X as OLE_XPOS_PIXELS	A single that specifies the current X location of the mouse pointer. The x values is always expressed in client coordinates.
Y as OLE_YPOS_PIXELS	A single that specifies the current Y location of the mouse pointer. The y values is always expressed in client coordinates.
Date	A Date expression that indicates the date from the cursor.

The DateFromPoint property indicates the date from the cursor in the calendar panel. The DateTimeFromPoint property indicates the date/time from the cursor on the schedule panel. The TimeFromPoint property indicates the time from the cursor on the schedule panel. You can use the Selection/SelCount/SelDate property to retrieve the selected dates. Use the Selection/SelectDate property to change programmatically the dates being selected in the calendar, including the dates to be shown in the schedule view. The FirstVisibleDate/LastVisibleDate property indicates the first visible date in the calendar panel.

The DateFromPoint property retrieves the value based on the X and Y parameters as follows:

• 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 /NET and /WPF versions provide a DateFromPoint property (with no arguments), that determines the date from the current mouse position, as the get_DateFromPoint(-1,-1) returns.

property Calendar.DisableZoneFormat as String

Returns or sets an expression that determines the dates being disabled in the calendar/schedule panel.

Туре	Description
String	A String expression that indicates the date to be disabled. The DisableZoneFormat property supports value,
	operators and predefined functions as listed bellow

By default, the DisableZoneFormat property is "", which means it has no effect. The DisableZoneFormat property may be used to specify the dates to be shown as disabled. The user can not update or create new events in a disabled zone, while the <u>AllowUpdateDisableZone</u> property is False (by default). The DisableZoneFormat property on "1" disables the entire schedule. A disabled zone always shows in gray. The <u>MinDate</u> property specifies the lower margin that the calendar panel could show. The <u>MaxDate</u> property specifies the upper margin that the calendar panel could show. The <u>AllowCreateEvent</u> property indicates the combination of the keys to let user creates new events. The <u>AllowMoveEvent</u> property indicates the combination of the keys to let user moves the events. The <u>AllowResizeEvent</u> property indicates the combination of the keys to let user moves the events. The <u>AllowResizeEvent</u> property indicates the combination of the keys to let user moves the events.

Here's a list of few samples:

- "1", disables all dates in the schedule view.
- "month(value) = 7" disables the dates in July
- "weekday(value) in (0,6)" disables the Sunday and Saturday
- "value in (#1/8/2001#,#1/9/2001#)" disables the 8 and 9 of January 2001
- "not (month(value) = month(value+1))" disables the last day of each month
- "(weekday(value) = 5) and not (month(value) = month(value+7))" disables the last Friday of each month
- "value < date(``)" disables all events that happened (the date(``) returns today, so all days before today)
- "value > date(``)" disables all events that will happen (the date(``) returns today, so all days after today)

The following screen shot shows all dates as disabled (the entire month is selected):

		May 1	2	3	4	5	6)^
May 2012								
Mo Tu We Th Fr Sa Su	7	8	9	10	11	12	13	
18 123456								
19 7 8 9 10 11 12 13 20 14 15 16 17 18 19 20	14]	16][]	18	19	20	
21 21 22 23 24 25 26 27								
22 28 29 30 31	21	22	23	24	25	26	27	
23								
	28	29	30	May 31				_
Today		=						-

DisableZoneFormat = "1"

A disable zone, shows as grayed as in the following screen shot (only dates: 23, 24, and 25 are enabled, and the rest are disabled, the entire month is selected):

		May 1	2	3	4	5	6	^
May 2012	7	8	9	10	11	12	13	Ĩ
18 1 2 3 4 5 6								
19 7 8 9 10 11 12 13	14	15	16	17	18	19	20	
20 14 15 16 17 18 19 20							·	
21 21 22 23 24 25 26 27 1 22 28 29 30 30	24				25	00	07	
23]]	23		25	20		
	28	29	30	May 31				
Today								-

DisableZoneFormat = "not day(value) in (23,24,25)"

The **value** keyword indicates the date to be disabled, and the predefined operators and functions are:

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#, #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
- Itrim (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)

property Calendar.DisplayWeekNumberAs as WeekNumberAsEnum

Specifies the way the control displays the week number.

Туре	Description
<u>WeekNumberAsEnum</u>	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 <u>ShowWeeks</u> property specifies whether the week number header is shown or hidden. The <u>FirstWeekDay</u> 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):

 ✓ January 2016 							
	М	т	W	т	F	S	S
53	28	29	30	31	1	2	3
1	4	5	6	7	8	9	10
2	50	12	13	14	15	16	17
3	18	19	20	21	22	23	24

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

 January 2016							
	S	М	Т	W	Т	F	S
1	27	28	29	30	31	1	2
2	71	4	5	6	7	8	9
3	10	11	12	13	14	15	16

property Calendar.Events ([Expression as Variant], [GroupID as Variant]) as Variant

Returns a safe array of dates with events, in a specified group, where the Expression indicates the formula to determine the dates being verified.

Туре	Description
Expression as Variant	A String expression that determines the dates being queried for events. The Expression parameter supports value, operators and predefined functions as listed bellow. If "1" is used, all dates within the browsed year is queried for events.
GroupID as Variant	A long expression that specifies identifier of the group to be queried. If missing, or -1, all events are searched, rather than events within specified group.
Variant	A safe array of DATE expressions (VT_ARRAY VT_VARIANT) , or a collection of DATEs that contain events.

The Events property gets a collection of DATEs that contain events. The Events property can be used to determine the dates with events. *The /NET and /WPF versions provide the get_EventsAs of the List<DateTime> that is an equivalent with the get_Events excepts that the returned type is a collection of DateTime objects.* The <u>DateEvents</u> property returns a collection of events in the specified date.

Here's a list of few samples:

- Events("1"), gets all dates with events in the browsed year
- Events("month(value) = 7") gets all dates with events from July
- Events("value = int(date(``))") gets a non-zero value if there is any event today
- Events("month(value) = month(int(date(``)))") gets the dates within the current month, that contains any appointments

The following samples gets the dates within the current month, that contains any appointments:

VB

```
Dim d As Variant
For Each d In Schedule1.Calendar.Events("month(value) = month(int(date(``)))")
Debug.Print d
Next
```

VB/NET

```
Dim evs As List(Of Date) = Exschedule1.Calendar.get_EventsAs("month(value) =
month(int(date(``)))")
If Not evs Is Nothing Then
For Each d As DateTime In evs
Debug.Print(d.ToString())
Next
End If
```

C#

```
List < DateTime > evs = exschedule1.Calendar.get_EventsAs("month(value) =
month(int(date(``)))");
if ( evs != null )
foreach (DateTime d in evs)
System.Diagnostics.Debug.Print(d.ToString());
```

VFP

```
local d, evs as Object
evs = thisform.Schedule1.ExecuteTemplate("Calendar.Events(" + CHR(34) +
"month(value)=month(date(``))" + CHR(34) +")")
For Each d In evs
WAIT WINDOW TTOC(d)
ENDFOR
```

C++

```
_variant_t selection = m_spSchedule->Calendar->Events["month(value) =
month(int(date(``)))"];
if ( V_VT( &selection ) == ( VT_ARRAY | VT_VARIANT ) )
{
    BYTE* p = NULL;
    long nCount = 0;
    if ( SUCCEEDED( SafeArrayGetUBound( V_ARRAY( &selection ), 1, &nCount ) ) )
    {
        if ( SUCCEEDED( SafeArrayAccessData( V_ARRAY( &selection ), (LPVOID*)&p ) ) )
        {
        }
        }
        }
    }
}
```

```
for ( long i = 0; i < nCount + 1; i++, p += sizeof(VARIANT) )
{
    VARIANT* pValue = (VARIANT*)p;
    if ( V_VT( pValue ) == VT_DATE )
    {
        CString strMessage;
        strMessage.Format(_T("%f\r\n"), V_DATE( pValue ) );
        OutputDebugString( strMessage );
    }
    SafeArrayUnaccessData( V_ARRAY( &selection ) );
}</pre>
```

where m_spSchedule is of EXSCHEDULELib::ISchedulePtr type.

The **value** keyword indicates the date being queried, and the predefined operators and functions are:

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#, #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:

- $\circ~$ 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
- $\circ~$ 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
- Itrim (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)
property Calendar.FirstVisibleDate as Date

Retrieves the first visible date, in the calendar panel.

Туре	Description
Date	A DATE expression that specifies the first visible date in the calendar panel.

The FirstVisibleDate property indicates the first visible date in the calendar panel. Use the <u>Date</u> property to browse a new date/month in the calendar panel. The <u>ShowNonMonthDays</u> property specifies whether the calendar panel displays the dates that are not part of the month.

property Calendar.FirstWeekDay as WeekDayEnum

Specifies the first day of the week.

Туре	Description
<u>WeekDayEnum</u>	A WeekDayEnum expression that specifies the first day of the week

By default, the FirstWeekDay property is exSunday. Use the <u>LocFirstWeekDay</u> property to get the locale first day of the week as indicated by current regional settings. The <u>AMPM</u> property indicates the AM, PM time indicators to be shown in the control, separated by space,. The <u>MonthNames</u> property specifies the list of name of the months. The <u>WeekDays</u> property specifies the name of the days in the week.

The following samples set the current view to display the locate date/time as set in the current regional settings.

VBA (MS Access, Excell...)

```
With Schedule1
With .Calendar
.FirstWeekDay = .LocFirstWeekDay
.MonthNames = .LocMonthNames
.WeekDays = .LocWeekDays
.AMPM = .LocAMPM
End With
End With
```

VB6

With Schedule1 With .Calendar .FirstWeekDay = .**LocFirstWeekDay** .MonthNames = .**LocMonthNames** .WeekDays = .**LocWeekDays** .AMPM = .**LocAMPM** End With End With

```
With Exschedule1
With .Calendar
.FirstWeekDay = .LocFirstWeekDay
.MonthNames = .LocMonthNames
.WeekDays = .LocWeekDays
.AMPM = .LocAMPM
End With
End With
```

VB.NET for /COM

With AxSchedule1 With .Calendar .FirstWeekDay = .**LocFirstWeekDay** .MonthNames = .**LocMonthNames** .WeekDays = .**LocWeekDays** .AMPM = .**LocAMPM** End With End With

C++

```
/*
   Copy and paste the following directives to your header file as
    it defines the namespace 'EXSCHEDULELib' for the library: 'ExSchedule 1.0 Control
Library'
   #import <ExSchedule.dll>
   using namespace EXSCHEDULELib;
   */
EXSCHEDULELib::ISchedulePtr spSchedule1 = GetDlgItem(IDC_SCHEDULE1)-
   >GetControlUnknown();
EXSCHEDULELib::ICalendarPtr var_Calendar = spSchedule1->GetCalendar();
   var_Calendar->PutFirstWeekDay(var_Calendar->GetLocFirstWeekDay());
   var_Calendar->PutMonthNames(var_Calendar->GetLocMonthNames());
   var_Calendar->PutWeekDays(var_Calendar->GetLocWeekDays());
   var_Calendar->PutAMPM(var_Calendar->GetLocAMPM());
```

C++ Builder

Exschedulelib_tlb::ICalendarPtr var_Calendar = Schedule1->Calendar; var_Calendar->FirstWeekDay = var_Calendar->LocFirstWeekDay; var_Calendar->MonthNames = var_Calendar->LocMonthNames; var_Calendar->WeekDays = var_Calendar->LocWeekDays; var_Calendar->AMPM = var_Calendar->LocAMPM;

C#

exontrol.EXSCHEDULELib.Calendar var_Calendar = exschedule1.Calendar; var_Calendar.FirstWeekDay = var_Calendar.LocFirstWeekDay; var_Calendar.MonthNames = var_Calendar.LocMonthNames; var_Calendar.WeekDays = var_Calendar.LocWeekDays; var_Calendar.AMPM = var_Calendar.LocAMPM;

JavaScript

```
<OBJECT classid="clsid:9B09E13D-7A88-4299-9DBE-383380435377"
id="Schedule1"></OBJECT>
</SCRIPT LANGUAGE="JScript">
var var_Calendar = Schedule1.Calendar;
var_Calendar.FirstWeekDay = var_Calendar.LocFirstWeekDay;
var_Calendar.MonthNames = var_Calendar.LocMonthNames;
var_Calendar.WeekDays = var_Calendar.LocWeekDays;
var_Calendar.AMPM = var_Calendar.LocAMPM;
</SCRIPT>
```

C# for /COM

EXSCHEDULELib.Calendar var_Calendar = axSchedule1.Calendar; var_Calendar.FirstWeekDay = var_Calendar.LocFirstWeekDay; var_Calendar.MonthNames = var_Calendar.LocMonthNames; var_Calendar.WeekDays = var_Calendar.LocWeekDays; var_Calendar.AMPM = var_Calendar.LocAMPM;

```
public void init()
{
    COM com_Calendar;
    anytype var_Calendar;
    ;
    super();
    var_Calendar = exschedule1.Calendar(); com_Calendar = var_Calendar;
    com_Calendar.FirstWeekDay(com_Calendar.LocFirstWeekDay());
    com_Calendar.MonthNames(com_Calendar.LocMonthNames());
    com_Calendar.WeekDays(com_Calendar.LocWeekDays());
    com_Calendar.AMPM(com_Calendar.LocAMPM());
}
```

Delphi 8 (.NET only)

```
with AxSchedule1 do
begin
with Calendar do
begin
FirstWeekDay := LocFirstWeekDay;
MonthNames := LocMonthNames;
WeekDays := LocWeekDays;
AMPM := LocAMPM;
end;
end
```

Delphi (standard)

```
with Schedule1 do
begin
with Calendar do
begin
FirstWeekDay := LocFirstWeekDay;
MonthNames := LocMonthNames;
```

```
WeekDays := LocWeekDays;
AMPM := LocAMPM;
end;
end
```

VFP

```
with thisform.Schedule1
with .Calendar
.FirstWeekDay = .LocFirstWeekDay
.MonthNames = .LocMonthNames
.WeekDays = .LocWeekDays
.AMPM = .LocAMPM
endwith
endwith
```

dBASE Plus

```
local oSchedule,var_Calendar
oSchedule = form.Activex1.nativeObject
var_Calendar = oSchedule.Calendar
```

var_Calendar.FirstWeekDay = var_Calendar.LocFirstWeekDay
var_Calendar.MonthNames = var_Calendar.LocMonthNames
var_Calendar.WeekDays = var_Calendar.LocWeekDays

```
var_Calendar.AMPM = var_Calendar.LocAMPM
```

XBasic (Alpha Five)

```
Dim oSchedule as P
Dim var_Calendar as P
oSchedule = topparent:CONTROL_ACTIVEX1.activex
var_Calendar = oSchedule.Calendar
var_Calendar.FirstWeekDay = var_Calendar.LocFirstWeekDay
var_Calendar.MonthNames = var_Calendar.LocMonthNames
var_Calendar.WeekDays = var_Calendar.LocWeekDays
```

```
var_Calendar.AMPM = var_Calendar.LocAMPM
```

Visual Objects

```
local var_Calendar as ICalendar
```

var_Calendar := oDCOCX_Exontrol1:Calendar
var_Calendar:FirstWeekDay := var_Calendar:LocFirstWeekDay
var_Calendar:MonthNames := var_Calendar:LocMonthNames
var_Calendar:WeekDays := var_Calendar:LocWeekDays
var_Calendar:AMPM := var_Calendar:LocAMPM

PowerBuilder

```
OleObject oSchedule,var_Calendar
oSchedule = ole_1.Object
var_Calendar = oSchedule.Calendar
var_Calendar.FirstWeekDay = var_Calendar.LocFirstWeekDay
var_Calendar.MonthNames = var_Calendar.LocMonthNames
var_Calendar.WeekDays = var_Calendar.LocWeekDays
var_Calendar.AMPM = var_Calendar.LocAMPM
```

property Calendar.FitSelToView as Variant

Specifies the list of additional dates to be shown on the schedule view, when OnSelectDate property is exFitSelToView.

Туре	Description
Variant	A single date or a safe array of dates to be included in the schedule chart when selection is changed.

By default, the FitSelToView property is empty. The FitSelToView property has effect only if the <u>OnSelectDate</u> property is exFitSelToView. The FitSelToView property specifies the list of additional dates (offset) to be included in the schedule view, when selection is changed. By default, the OnSelectDate property is exFitSelToView, which indicates that the selected date in the calendar panel, is enlarged so it fit the schedule view. For instance, your chart need to display not just the selected date, but to include also one day before and after, so you need to set the FitSelToView on Array(-1,1). Use the <u>Selection</u> property to select dates in the calendar panel.

The following VB samples ensures that the schedule-view displays at least 3 days (a day before selected date, the selected date , and a day after the selected date) when the user selects a date in the calendar panel:

```
With Schedule1.Calendar
.OnSelectDate = exFitSelToView
.FitSelToView = Array(-1, 1)
End With
```

In other words, use the FitSelToView property to specify the dates to be included in the schedule view when user changes the selection in the calendar panel.

property Calendar.FocusDate as Date

Retrieves the date being focused in the calendar panel.

Туре	Description
Date	A DATE expression being focused in the calendar panel.

The FocusDate property indicates the date being focused. The control fires the LayoutStartChanging(exCalendarFocusDateChange) when the user is about to change the focusing date, and the LayoutEndChanging(exCalendarFocusDateChange) notifies your application once a new date is being focused. The Background(exCalendarFocusDate) changes the visual appearance of the focused date, while the Background(exCalendarFocusDateForeColor) changes the foreground color of the focused date. The control may display only a single focused date, but it can display multiple selection dates. The AllowFocusDate property on exDisallow disables focusing a date that's not being selected. The AllowSelectDate property indicates the keys combination so the user can select new dates in the calendar panel, and so, new dates to be shown in the schedule view. The Date property of the control browses a new month in the calendar panel. The Select method can be used to select by code the current month, current week, current week day and the current/focus day.

property Calendar.GroupHighlightEvent as Boolean

Highlights the date in the calendar panel using the CalendarHighlightEvent property of each Group found on day's events.

Туре	Description
Boolean	A Boolean expression that specifies whether the dates in the calendar panel using the Calendar HighlightEvent property of each Group found on day's events

By default, the GroupHighlightEvent property is False. The GroupHighlightEvent property specifies if events are highlighted using the <u>HighlightEvent</u> property (False), or using the <u>CalendarHighlightEvent</u> property of the <u>Group</u> that event belongs to (True). The <u>ShowHighlightEvent</u> property specifies whether the calendar panel highlights the events in the calendar panel. The <u>ScheduleHighlightEvent</u> property specifies the visual appearance of dates with events in the schedule panel. The <u>GroupID</u> property indicates the identifier of the event's group.

The following screen shot shows the dates with events when GroupHighlightEvent property is True:



property Calendar.HeaderDayLabel as String

Specifies the HTML date-format to be shown on the calendar's header.

Туре	Description
String	A String expression that specifies the HTML date-format to be shown on the calendar's header.

By default, the HeaderDayLabel property is "<%mmm%> <%yyyy%>", which shows the month and the year in the header of the calendar panel. The HeaderDayLongLabel/HeaderDayShortLabel properties specifies the HTML date-format to be displayed on the schedule view (by default, the right side panel). The Background(exCalendarHeader) specifies the visual appearance of the calendar's header. The Background(exCalendarHeaderForeColor) specifies the foreground color of the calendar's header.

The following screen shot shows the calendar's header using the format "<sha> <%mmm%></sha> <sha><fgcolor=FF0000><%yyyy%></fgcolor></sha>"



The following screen shot shows the calendar's header (by default):

4	August 2012							
	29	30	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	31	1	
	2	3	4	5	6	7	8	

The property supports the following built-in HTML tags:

- ... displays the text in **bold**
- <i> ... </i> displays the text in *italics*
- <u> ... </u> <u>underlines</u> the text
- <s> ... </s> Strike-through text

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

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

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

Any ex-HTML caption can be transformed to an expandable-caption, by inserting the anchor ex-HTML tag. For instance, "<solidline>Header</solidline>

Line1<r><a ;exp=show lines>+
Line2
Line3" shows the Header in underlined and bold on the first line and Line1, Line2, Line3 on the rest. The "show lines" is shown instead of Line1, Line2, Line3 once the user clicks the + sign.

- ... displays portions of text with a different font and/or different size. For instance, the "bit" draws the bit text using the Tahoma font, on size 12 pt. If the name of the font is missing, and instead size is present, the current font is used with a different size. For instance, "bit" displays the bit text using the current font, but with a different size.
- **<fgcolor rrggbb>** ... **</fgcolor>** or **<**fgcolor=rrggbb> ... **</fgcolor>** displays text with a specified foreground color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- <bgcolor rrggbb> ... </bgcolor> or <bgcolor=rrggbb> ... </bgcolor> displays text with a specified background color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- <solidline rrggbb> ... </solidline> or <solidline=rrggbb> ... </solidline> draws a solidline 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
- number[:width] 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.
- key[:width] 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 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

FFFFF;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:



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

shadow

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

outline anti-aliasing

The property supports the following TAGs:

- <%d%> Day of the month in one or two numeric digits, as needed (1 to 31).
- <%dd%> Day of the month in two numeric digits (01 to 31).
- <%d1%> First letter of the weekday (S to S). (Use the <u>WeekDays</u> property to specify the name of the days in the week)
- <%d2%> First two letters of the weekday (Su to Sa). (Use the <u>WeekDays</u> property to specify the name of the days in the week)
- <%d3%> First three letters of the weekday (Sun to Sat). (Use the <u>WeekDays</u> property to specify the name of the days in the week)
- <%ddd%> First three letters of the weekday (Sun to Sat). (Use the <u>WeekDays</u> property to specify the name of the days in the week). You can use the <%loc_ddd%> that indicates the day of week as a three-letter abbreviation using the current user regional and language settings.
- <%loc_ddd%> Indicates the day of week as a three-letter abbreviation using the current user regional and language settings.

- <%dddd%> Full name of the weekday (Sunday to Saturday). (Use the <u>WeekDays</u> property to specify the name of the days in the week). You can use the <%loc_dddd%> that indicates day of week as its full name using the current user regional and language settings.
- <%loc_dddd%> Indicates day of week as its full name using the current user regional and language settings.
- <%i%> Displays the number instead the date. For instance, you can display numbers as 1000, 1001, 1002, 1003, instead dates. (the valid range is from -647,434 to 2,958,465)
- <%**w**%> Day of the week (1 to 7).
- <%**ww**%> Week of the year (1 to 53).
- <%m%> Month of the year in one or two numeric digits, as needed (1 to 12).
- <%mr%> Month of the year in Roman numerals, as needed (I to XII).
- <%mm%> Month of the year in two numeric digits (01 to 12).
- <%m1%> First letter of the month (J to D). (Use the <u>MonthNames</u> property to specify the name of the months in the year)
- <%m2%> First two letters of the month (Ja to De). (Use the <u>MonthNames</u> property to specify the name of the months in the year)
- <%m3%> First three letters of the month (Jan to Dec). (Use the <u>MonthNames</u> property to specify the name of the months in the year)
- <%mmm%> First three letters of the month (Jan to Dec). (Use the <u>MonthNames</u> property to specify the name of the months in the year). You can use the <%loc_mmm%> that indicates month as a three-letter abbreviation using the current user regional and language settings.
- <%loc_mmm%> Indicates month as a three-letter abbreviation using the current user regional and language settings.
- <%mmm%> Full name of the month (January to December). (Use the MonthNames property to specify the name of the months in the year). You can use the <%loc_mmm%> that indicates month as its full name using the current user regional and language settings.
- <%loc_mmmm%> Indicates month as its full name using the current user regional and language settings.
- <%q%> Date displayed as the quarter of the year (1 to 4).
- $\langle \mathbf{y} \rangle$ Number of the day of the year (1 to 366).
- <%**yy**%> Last two digits of the year (01 to 99).
- <%**yyyy**%> Full year (0100 to 9999).
- <%hy%> Date displayed as the half of the year (1 to 2).
- <%loc_gg%> Indicates period/era using the current user regional and language settings.
- <%loc_sdate%> Indicates the date in the short format using the current user regional and language settings.
- <%loc_ldate%> Indicates the date in the long format using the current user regional

and language settings.

- <%loc_dsep%> Indicates the date separator using the current user regional and language settings (/).
- <%h%> Hour in one or two digits, as needed (0 to 23).
- <%**hh**%> Hour in two digits (00 to 23).
- <%**n**%> Minute in one or two digits, as needed (0 to 59).
- <%nn%> Minute in two digits (00 to 59).
- <%s%> Second in one or two digits, as needed (0 to 59).
- <%**ss**%> Second in two digits (00 to 59).
- <%AM/PM%> Twelve-hour clock with the uppercase letters "AM" or "PM", as appropriate. (Use the <u>AMPM</u> property to specify the name of the AM and PM indicators). You can use the <%loc_AM/PM%> that indicates the time marker such as AM or PM using the current user regional and language settings. You can use <%loc_A/P%> that indicates the one character time marker such as A or P using the current user regional and language settings.
- <%loc_AM/PM%> Indicates the time marker such as AM or PM using the current user regional and language settings.
- <%loc_A/P%> Indicates the one character time marker such as A or P using the current user regional and language settings.
- <%loc_time%> Indicates the time using the current user regional and language settings.
- <%loc_time24%> Indicates the time in 24 hours format without a time marker using the current user regional and language settings.
- <%loc_tsep%> indicates the time separator using the current user regional and language settings (:).
- <%loc_y%> Represents the Year only by the last digit, using current regional settings.
- <%loc_yy%> Represents the Year only by the last two digits, using current regional settings. A leading zero is added for single-digit years.
- <%loc_yyyy%> Represents the Year by a full four or five digits, depending on the calendar used. Thai Buddhist and Korean calendars have five-digit years. The "yyy" pattern shows five digits for these two calendars, and four digits for all other supported calendars. Calendars that have single-digit or two-digit years, such as for the Japanese Emperor era, are represented differently. A single-digit year is represented with a leading zero, for example, "03". A two-digit year is represented with two digits, for example, "13". No additional leading zeros are displayed.

The following tags are displayed based on the user's Regional and Language Options:

- <%loc_sdate%> Indicates the date in the short format using the current user settings.
- <%loc_ldate%> Indicates the date in the long format using the current user settings.

- <%loc_ddd%> Indicates day of week as a three-letter abbreviation using the current user settings.
- <%loc_dddd%> Indicates day of week as its full name using the current user settings.
- <%loc_mmm%> Indicates month as a three-letter abbreviation using the current user settings.
- <%loc_mmmm%> Indicates month as its full name using the current user settings.
- <%loc_gg%> Indicates period/era using the current user settings.
- <%loc_dsep%> Indicates the date separator using the current user settings.
- <%loc_time%> Indicates the time using the current user settings.
- <%loc_time24%> Indicates the time in 24 hours format without a time marker using the current user settings.
- <%loc_AM/PM%> Indicates the time marker such as AM or PM using the current user settings.
- <%loc_A/P%> Indicates the one character time marker such as A or P using the current user settings.
- <%loc_tsep%> Indicates the time separator using the current user settings.
- <%loc_y%> Represents the Year only by the last digit, using current regional settings.
- <%loc_yy%> Represents the Year only by the last two digits, using current regional settings. A leading zero is added for single-digit years.
- <%loc_yyyy%> Represents the Year by a full four or five digits, depending on the calendar used. Thai Buddhist and Korean calendars have five-digit years. The "yyyy" pattern shows five digits for these two calendars, and four digits for all other supported calendars. Calendars that have single-digit or two-digit years, such as for the Japanese Emperor era, are represented differently. A single-digit year is represented with a leading zero, for example, "03". A two-digit year is represented with two digits, for example, "13". No additional leading zeros are displayed.

property Calendar.HideSel as Boolean

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

Туре	Description
Boolean	A Boolean expression that specifies whether the selected dates shows as selected when the control loses the focus.

By default, the HideSel property is False, which means that the selected dates are still shown when the control loses the focus. Use the HideSel property to show no selected dates in the calendar panel, when the component loses the focus. If the HideSel property is False (by default), the Background(exCalendarSelBackColorUnFocus) specifies the visual appearance of the selected dates when the component is not focused. The Background(exCalendarSelForeColorUnFocus) property indicates the foreground color of the selected dates when the control loses the focus. The FocusDate property indicates the DATE in the calendar panel which is focused.

The <u>AllowSelectDate</u> property indicates the keys combination so the user can select new dates in the calendar panel, and so, new dates to be shown in the schedule view. Use the <u>Selection/SelectDate</u> property to change programmatically the dates being selected in the calendar, including the dates to be shown in the schedule view. You can use the <u>Selection/SelCount/SelDate</u> property to retrieve the selected dates.

property Calendar.HighlightEvent as Highlight

Gives access to the Highlight object, so you can customize highlighting the events, in the calendar panel.

TypeDescriptionHighlightA Highlight object to customize the dates with events.

By default, the dates with events or appointments appear as bold in the calendar panel. You can use the HighlightEvent object to highlight the dates with events in the calendar panel. The <u>ShowHighlightEvent</u> property specifies whether the calendar panel highlights the events in the calendar panel. The <u>GroupHighlightEvent</u> property specifies if events are highlighted using the HighlightEvent property (False), or using the <u>CalendarHighlightEvent</u> property of the <u>Group</u> that event belongs to (True). The <u>ScheduleHighlightEvent</u> property specifies the visual appearance of dates with events in the schedule panel.

Using the Highlight object a date with events can combine one or more of the following options:

- **bold**, **<u>Bold</u> property renders as bold text**
- *italic*, <u>Italic</u> property renders as italic text
- <u>underline</u>, <u>Underline</u> property underlines the text
- strikeout, StrikeOut property shows the text with a horizontal line through its center
- change the font SiZe, <u>FontSize</u> property indicates the size of the font to display the text
- change the **font**, using the **Font** property
- change the text's foreground color, using the ForeColor property
- change the text's background color, using the BackColor property
- shows a pattern using the <u>Pattern</u> property

The following screen shot shows the dates with events using a frame around:

		Thursday, May 17, 2012	Friday,May 18,2012
May 2012 → Mo Tu We Th Fr Sa Su 1 1 2 3 4 5 6 19 7 8 9 10 11 12 13	08:00 AM :30 09:00 AM :30 10:00 AM :30 11:00 AM	9:00 AM - 11:15 AM 10:00 AM - 11:30 AM	
20 14 15 16 7 19 20 21 21 22 23 24 25 26 27 22 28 29 30 31 23	:30 12:00 PM :30 01:00 PM :30	12:15 PM - 2:00 PM no title	
Today	02:00 PM :30 03:00 PM :30		no title

The following screen shot shows the dates with events when <u>GroupHighlightEvent</u> property is True:

									Thursday,Ma	ay 17,2012	Fr	iday,May	18,2012	1
4			May	2012	2		•	08:00 AM :30	7 Prima	1 Pro	7	Prima	1 Pro	
	Мо	Tu	We	Th	Fr	Sa	Su	09:00 AM :30	9:00 AM - 11:15 AM		9:15 A	AM -		
18		1	2	3	4	5	6	10:00 AM		10:00 AM -	12:00	PM		
19	7	8	9	10	11	12	13	:30	-	11:30 AM				
20	14	15	16	17	18	19	20	11:00 AM :30						
21	21	22	23	24	25	26	27	12:00 PM	12:15 DM	12:00 PM -				
22	28	29	30	31				:30	2:00 PM	2:00 PM	12			
23								:30			2:45			
	¢							02:00 PM :30 03:00 PM :30	•		PM	1:45 PM - 3:45 PM		
			Too	lay]			٠ ١			1			

property Calendar.hWnd as Long

Retrieves the calendar's window handle.

l

Туре	Description
Long	A long expression that indicates the calendar's window handle.

Use the hWnd property to get the handle of the calendar panel. Use the <u>hWnd</u> property to get the handle of the control. 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. Use the <u>Parent</u> property to host the calendar panel by another window.

property Calendar.LastVisibleDate as Date

Retrieves the last visible date, in the calendar panel.

Туре	Description
Date	A DATE expression that specifies the last visible date in the calendar panel.

The LastVisibleDate property indicates the last visible date in the calendar panel. Use the <u>Date</u> property to browse a new date/month in the calendar panel. The <u>ShowNonMonthDays</u> property specifies whether the calendar panel displays the dates that are not part of the month.

property Calendar.LocAMPM as String

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

Туре	Description
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 <%AM/PM%> HTML tag indicates the twelve-hour clock with the uppercase letters "AM" or "PM", as appropriate set by the AMPM property. The <%loc_AM/PM%> HTML tag indicates the time marker such as AM or PM using the current user regional and language settings (LocAMPM property). The LocFirstWeekDay 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 following samples set the current view to display the locate date/time as set in the current regional settings.

VBA (MS Access, Excell...)

```
With Schedule1
With .Calendar
.FirstWeekDay = .LocFirstWeekDay
.MonthNames = .LocMonthNames
.WeekDays = .LocWeekDays
.AMPM = .LocAMPM
End With
```

VB6

With Schedule1 With .Calendar .FirstWeekDay = .**LocFirstWeekDay** .MonthNames = .**LocMonthNames** .WeekDays = .**LocWeekDays**

```
.AMPM = .LocAMPM
End With
End With
```

VB.NET

With Exschedule1 With .Calendar .FirstWeekDay = .**LocFirstWeekDay** .MonthNames = .**LocMonthNames** .WeekDays = .**LocWeekDays** .AMPM = .**LocAMPM** End With End With

VB.NET for /COM

With AxSchedule1 With .Calendar .FirstWeekDay = .**LocFirstWeekDay** .MonthNames = .**LocMonthNames** .WeekDays = .**LocWeekDays** .AMPM = .**LocAMPM** End With End With

C++

```
/*
Copy and paste the following directives to your header file as
    it defines the namespace 'EXSCHEDULELib' for the library: 'ExSchedule 1.0 Control
Library'
#import <ExSchedule.dll>
    using namespace EXSCHEDULELib;
*/
EXSCHEDULELib::ISchedulePtr spSchedule1 = GetDlgItem(IDC_SCHEDULE1)-
>GetControlUnknown();
```

```
EXSCHEDULELib::ICalendarPtr var_Calendar = spSchedule1->GetCalendar();
var_Calendar->PutFirstWeekDay(var_Calendar->GetLocFirstWeekDay());
var_Calendar->PutMonthNames(var_Calendar->GetLocMonthNames());
var_Calendar->PutWeekDays(var_Calendar->GetLocWeekDays());
var_Calendar->PutAMPM(var_Calendar->GetLocAMPM());
```

C++ Builder

```
Exschedulelib_tlb::ICalendarPtr var_Calendar = Schedule1->Calendar;
var_Calendar->FirstWeekDay = var_Calendar->LocFirstWeekDay;
var_Calendar->MonthNames = var_Calendar->LocMonthNames;
var_Calendar->WeekDays = var_Calendar->LocWeekDays;
var_Calendar->AMPM = var_Calendar->LocAMPM;
```

C#

```
exontrol.EXSCHEDULELib.Calendar var_Calendar = exschedule1.Calendar;
var_Calendar.FirstWeekDay = var_Calendar.LocFirstWeekDay;
var_Calendar.MonthNames = var_Calendar.LocMonthNames;
var_Calendar.WeekDays = var_Calendar.LocWeekDays;
var_Calendar.AMPM = var_Calendar.LocAMPM;
```

JavaScript

```
<OBJECT classid="clsid:9B09E13D-7A88-4299-9DBE-383380435377"
id="Schedule1"></OBJECT>
<SCRIPT LANGUAGE="JScript">
var var_Calendar = Schedule1.Calendar;
var_Calendar.FirstWeekDay = var_Calendar.LocFirstWeekDay;
var_Calendar.MonthNames = var_Calendar.LocMonthNames;
var_Calendar.WeekDays = var_Calendar.LocWeekDays;
var_Calendar.AMPM = var_Calendar.LocAMPM;
</SCRIPT>
```

C# for /COM

```
EXSCHEDULELib.Calendar var_Calendar = axSchedule1.Calendar;
var_Calendar.FirstWeekDay = var_Calendar.LocFirstWeekDay;
var_Calendar.MonthNames = var_Calendar.LocMonthNames;
var_Calendar.WeekDays = var_Calendar.LocWeekDays;
var_Calendar.AMPM = var_Calendar.LocAMPM;
```

X++ (Dynamics Ax 2009)

```
public void init()
{
    COM com_Calendar;
    anytype var_Calendar;
    ;
    super();
    var_Calendar = exschedule1.Calendar(); com_Calendar = var_Calendar;
        com_Calendar.FirstWeekDay(com_Calendar.LocFirstWeekDay());
        com_Calendar.MonthNames(com_Calendar.LocMonthNames());
        com_Calendar.WeekDays(com_Calendar.LocWeekDays());
        com_Calendar.WeekDays(com_Calendar.LocWeekDays());
        com_Calendar.AMPM(com_Calendar.LocAMPM());
    }
}
```

Delphi 8 (.NET only)

```
with AxSchedule1 do
begin
with Calendar do
begin
FirstWeekDay := LocFirstWeekDay;
MonthNames := LocMonthNames;
WeekDays := LocWeekDays;
AMPM := LocAMPM;
end;
end
```

Delphi (standard)

```
with Schedule1 do
begin
with Calendar do
begin
FirstWeekDay := LocFirstWeekDay;
MonthNames := LocMonthNames;
WeekDays := LocWeekDays;
AMPM := LocAMPM;
end;
end
```

VFP

```
with thisform.Schedule1
with .Calendar
.FirstWeekDay = .LocFirstWeekDay
.MonthNames = .LocMonthNames
.WeekDays = .LocWeekDays
.AMPM = .LocAMPM
endwith
endwith
```

dBASE Plus

```
local oSchedule,var_Calendar
```

```
oSchedule = form.Activex1.nativeObject
var_Calendar = oSchedule.Calendar
var_Calendar.FirstWeekDay = var_Calendar.LocFirstWeekDay
var_Calendar.MonthNames = var_Calendar.LocMonthNames
var_Calendar.WeekDays = var_Calendar.LocWeekDays
var_Calendar.AMPM = var_Calendar.LocAMPM
```

XBasic (Alpha Five)

Dim oSchedule as P Dim var_Calendar as P

```
oSchedule = topparent:CONTROL_ACTIVEX1.activex
var_Calendar = oSchedule.Calendar
var_Calendar.FirstWeekDay = var_Calendar.LocFirstWeekDay
var_Calendar.MonthNames = var_Calendar.LocMonthNames
var_Calendar.WeekDays = var_Calendar.LocWeekDays
var_Calendar.AMPM = var_Calendar.LocAMPM
```

Visual Objects

local var_Calendar as ICalendar

var_Calendar := oDCOCX_Exontrol1:Calendar
var_Calendar:FirstWeekDay := var_Calendar:LocFirstWeekDay
var_Calendar:MonthNames := var_Calendar:LocMonthNames
var_Calendar:WeekDays := var_Calendar:LocWeekDays
var_Calendar:AMPM := var_Calendar:LocAMPM

PowerBuilder

OleObject oSchedule,var_Calendar oSchedule = ole_1.Object var_Calendar = oSchedule.Calendar var_Calendar.FirstWeekDay = var_Calendar.LocFirstWeekDay var_Calendar.MonthNames = var_Calendar.LocMonthNames var_Calendar.WeekDays = var_Calendar.LocWeekDays var_Calendar.AMPM = var_Calendar.LocAMPM

property Calendar.LocFirstWeekDay as WeekDayEnum

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

Туре	Description
<u>WeekDayEnum</u>	A WeekDayEnum expression that specifies the first day of the week, as specified in the regional settings.

The LocFirstWeekDay 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.

The following samples set the current view to display the locate date/time as set in the current regional settings.

VBA (MS Access, Excell...)

```
With Schedule1
With .Calendar
.FirstWeekDay = .LocFirstWeekDay
.MonthNames = .LocMonthNames
.WeekDays = .LocWeekDays
.AMPM = .LocAMPM
End With
End With
```

VB6

With Schedule1 With .Calendar .FirstWeekDay = .**LocFirstWeekDay** .MonthNames = .**LocMonthNames** .WeekDays = .**LocWeekDays** .AMPM = .**LocAMPM** End With End With

```
With Exschedule1
With .Calendar
.FirstWeekDay = .LocFirstWeekDay
.MonthNames = .LocMonthNames
.WeekDays = .LocWeekDays
.AMPM = .LocAMPM
End With
End With
```

VB.NET for /COM

With AxSchedule1 With .Calendar .FirstWeekDay = .**LocFirstWeekDay** .MonthNames = .**LocMonthNames** .WeekDays = .**LocWeekDays** .AMPM = .**LocAMPM** End With End With

C++

```
/*
   Copy and paste the following directives to your header file as
    it defines the namespace 'EXSCHEDULELib' for the library: 'ExSchedule 1.0 Control
Library'
   #import <ExSchedule.dll>
   using namespace EXSCHEDULELib;
   */
EXSCHEDULELib::ISchedulePtr spSchedule1 = GetDlgItem(IDC_SCHEDULE1)-
   >GetControlUnknown();
EXSCHEDULELib::ICalendarPtr var_Calendar = spSchedule1->GetCalendar();
   var_Calendar->PutFirstWeekDay(var_Calendar->GetLocFirstWeekDay());
   var_Calendar->PutMonthNames(var_Calendar->GetLocMonthNames());
   var_Calendar->PutWeekDays(var_Calendar->GetLocWeekDays());
   var_Calendar->PutAMPM(var_Calendar->GetLocAMPM());
```

C++ Builder

Exschedulelib_tlb::ICalendarPtr var_Calendar = Schedule1->Calendar; var_Calendar->FirstWeekDay = var_Calendar->LocFirstWeekDay; var_Calendar->MonthNames = var_Calendar->LocMonthNames; var_Calendar->WeekDays = var_Calendar->LocWeekDays; var_Calendar->AMPM = var_Calendar->LocAMPM;

C#

exontrol.EXSCHEDULELib.Calendar var_Calendar = exschedule1.Calendar; var_Calendar.FirstWeekDay = var_Calendar.LocFirstWeekDay; var_Calendar.MonthNames = var_Calendar.LocMonthNames; var_Calendar.WeekDays = var_Calendar.LocWeekDays; var_Calendar.AMPM = var_Calendar.LocAMPM;

JavaScript

```
<OBJECT classid="clsid:9B09E13D-7A88-4299-9DBE-383380435377"
id="Schedule1"></OBJECT>
</SCRIPT LANGUAGE="JScript">
var var_Calendar = Schedule1.Calendar;
var_Calendar.FirstWeekDay = var_Calendar.LocFirstWeekDay;
var_Calendar.MonthNames = var_Calendar.LocMonthNames;
var_Calendar.WeekDays = var_Calendar.LocWeekDays;
var_Calendar.AMPM = var_Calendar.LocAMPM;
</SCRIPT>
```

C# for /COM

EXSCHEDULELib.Calendar var_Calendar = axSchedule1.Calendar; var_Calendar.FirstWeekDay = var_Calendar.LocFirstWeekDay; var_Calendar.MonthNames = var_Calendar.LocMonthNames; var_Calendar.WeekDays = var_Calendar.LocWeekDays; var_Calendar.AMPM = var_Calendar.LocAMPM;

```
public void init()
{
    COM com_Calendar;
    anytype var_Calendar;
    ;
    super();
    var_Calendar = exschedule1.Calendar(); com_Calendar = var_Calendar;
    com_Calendar.FirstWeekDay(com_Calendar.LocFirstWeekDay());
    com_Calendar.MonthNames(com_Calendar.LocMonthNames());
    com_Calendar.WeekDays(com_Calendar.LocWeekDays());
    com_Calendar.AMPM(com_Calendar.LocAMPM());
}
```

Delphi 8 (.NET only)

```
with AxSchedule1 do
begin
with Calendar do
begin
FirstWeekDay := LocFirstWeekDay;
MonthNames := LocMonthNames;
WeekDays := LocWeekDays;
AMPM := LocAMPM;
end;
end
```

Delphi (standard)

```
with Schedule1 do
begin
with Calendar do
begin
FirstWeekDay := LocFirstWeekDay;
MonthNames := LocMonthNames;
```

```
WeekDays := LocWeekDays;
AMPM := LocAMPM;
end;
end
```

VFP

```
with thisform.Schedule1
with .Calendar
.FirstWeekDay = .LocFirstWeekDay
.MonthNames = .LocMonthNames
.WeekDays = .LocWeekDays
.AMPM = .LocAMPM
endwith
endwith
```

dBASE Plus

```
local oSchedule,var_Calendar
oSchedule = form.Activex1.nativeObject
var_Calendar = oSchedule.Calendar
```

var_Calendar.FirstWeekDay = var_Calendar.LocFirstWeekDay
var_Calendar.MonthNames = var_Calendar.LocMonthNames
var_Calendar.WeekDays = var_Calendar.LocWeekDays

```
var_Calendar.AMPM = var_Calendar.LocAMPM
```

XBasic (Alpha Five)

```
Dim oSchedule as P
Dim var_Calendar as P
oSchedule = topparent:CONTROL_ACTIVEX1.activex
var_Calendar = oSchedule.Calendar
var_Calendar.FirstWeekDay = var_Calendar.LocFirstWeekDay
var_Calendar.MonthNames = var_Calendar.LocMonthNames
var_Calendar.WeekDays = var_Calendar.LocWeekDays
```

```
var_Calendar.AMPM = var_Calendar.LocAMPM
```

Visual Objects

```
local var_Calendar as ICalendar
```

var_Calendar := oDCOCX_Exontrol1:Calendar
var_Calendar:FirstWeekDay := var_Calendar:LocFirstWeekDay
var_Calendar:MonthNames := var_Calendar:LocMonthNames
var_Calendar:WeekDays := var_Calendar:LocWeekDays
var_Calendar:AMPM := var_Calendar:LocAMPM

PowerBuilder

```
OleObject oSchedule,var_Calendar

oSchedule = ole_1.Object

var_Calendar = oSchedule.Calendar

var_Calendar.FirstWeekDay = var_Calendar.LocFirstWeekDay

var_Calendar.MonthNames = var_Calendar.LocMonthNames

var_Calendar.WeekDays = var_Calendar.LocWeekDays

var_Calendar.AMPM = var_Calendar.LocAMPM
```

property Calendar.LocMonthNames as String

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

Туре	Description
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 <%m1%>, <%m2%>, <%m3%>, <%mmm%> HTML tags indicate the name of the month, as appropriate set by the MonthNames property. The <%loc_m1%>, <%loc_m2%>, <%loc_m3%>, <%loc_mmm%> HTML tags indicate the month using the current user regional and language settings (LocMonthNames property). The LocFirstWeekDay 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 name of the days in the week, as indicated in the regional settings.

The following samples set the current view to display the locate date/time as set in the current regional settings.

VBA (MS Access, Excell...)

```
With Schedule1
With .Calendar
.FirstWeekDay = .LocFirstWeekDay
.MonthNames = .LocMonthNames
.WeekDays = .LocWeekDays
.AMPM = .LocAMPM
End With
```

VB6

With Schedule1 With .Calendar .FirstWeekDay = .**LocFirstWeekDay** .MonthNames = .**LocMonthNames** .WeekDays = .**LocWeekDays**

```
.AMPM = .LocAMPM
End With
End With
```

VB.NET

With Exschedule1 With .Calendar .FirstWeekDay = .**LocFirstWeekDay** .MonthNames = .**LocMonthNames** .WeekDays = .**LocWeekDays** .AMPM = .**LocAMPM** End With End With

VB.NET for /COM

With AxSchedule1 With .Calendar .FirstWeekDay = .**LocFirstWeekDay** .MonthNames = .**LocMonthNames** .WeekDays = .**LocWeekDays** .AMPM = .**LocAMPM** End With End With

C++

```
/*
Copy and paste the following directives to your header file as
    it defines the namespace 'EXSCHEDULELib' for the library: 'ExSchedule 1.0 Control
Library'
#import <ExSchedule.dll>
    using namespace EXSCHEDULELib;
*/
EXSCHEDULELib::ISchedulePtr spSchedule1 = GetDlgItem(IDC_SCHEDULE1)-
>GetControlUnknown();
```
```
EXSCHEDULELib::ICalendarPtr var_Calendar = spSchedule1->GetCalendar();
var_Calendar->PutFirstWeekDay(var_Calendar->GetLocFirstWeekDay());
var_Calendar->PutMonthNames(var_Calendar->GetLocMonthNames());
var_Calendar->PutWeekDays(var_Calendar->GetLocWeekDays());
var_Calendar->PutAMPM(var_Calendar->GetLocAMPM());
```

C++ Builder

```
Exschedulelib_tlb::ICalendarPtr var_Calendar = Schedule1->Calendar;
var_Calendar->FirstWeekDay = var_Calendar->LocFirstWeekDay;
var_Calendar->MonthNames = var_Calendar->LocMonthNames;
var_Calendar->WeekDays = var_Calendar->LocWeekDays;
var_Calendar->AMPM = var_Calendar->LocAMPM;
```

C#

```
exontrol.EXSCHEDULELib.Calendar var_Calendar = exschedule1.Calendar;
var_Calendar.FirstWeekDay = var_Calendar.LocFirstWeekDay;
var_Calendar.MonthNames = var_Calendar.LocMonthNames;
var_Calendar.WeekDays = var_Calendar.LocWeekDays;
var_Calendar.AMPM = var_Calendar.LocAMPM;
```

JavaScript

```
<OBJECT classid="clsid:9B09E13D-7A88-4299-9DBE-383380435377"
id="Schedule1"></OBJECT>
<SCRIPT LANGUAGE="JScript">
var var_Calendar = Schedule1.Calendar;
var_Calendar.FirstWeekDay = var_Calendar.LocFirstWeekDay;
var_Calendar.MonthNames = var_Calendar.LocMonthNames;
var_Calendar.WeekDays = var_Calendar.LocWeekDays;
var_Calendar.AMPM = var_Calendar.LocAMPM;
</SCRIPT>
```

C# for /COM

```
EXSCHEDULELib.Calendar var_Calendar = axSchedule1.Calendar;
var_Calendar.FirstWeekDay = var_Calendar.LocFirstWeekDay;
var_Calendar.MonthNames = var_Calendar.LocMonthNames;
var_Calendar.WeekDays = var_Calendar.LocWeekDays;
var_Calendar.AMPM = var_Calendar.LocAMPM;
```

X++ (Dynamics Ax 2009)

```
public void init()
{
    COM com_Calendar;
    anytype var_Calendar;
    ;
    super();
    var_Calendar = exschedule1.Calendar(); com_Calendar = var_Calendar;
        com_Calendar.FirstWeekDay(com_Calendar.LocFirstWeekDay());
        com_Calendar.MonthNames(com_Calendar.LocMonthNames());
        com_Calendar.WeekDays(com_Calendar.LocWeekDays());
        com_Calendar.WeekDays(com_Calendar.LocWeekDays());
        com_Calendar.AMPM(com_Calendar.LocAMPM());
    }
}
```

Delphi 8 (.NET only)

```
with AxSchedule1 do
begin
with Calendar do
begin
FirstWeekDay := LocFirstWeekDay;
MonthNames := LocMonthNames;
WeekDays := LocWeekDays;
AMPM := LocAMPM;
end;
end
```

Delphi (standard)

```
with Schedule1 do
begin
with Calendar do
begin
FirstWeekDay := LocFirstWeekDay;
MonthNames := LocMonthNames;
WeekDays := LocWeekDays;
AMPM := LocAMPM;
end;
end
```

VFP

```
with thisform.Schedule1
with .Calendar
.FirstWeekDay = .LocFirstWeekDay
.MonthNames = .LocMonthNames
.WeekDays = .LocWeekDays
.AMPM = .LocAMPM
endwith
endwith
```

dBASE Plus

```
local oSchedule,var_Calendar
```

```
oSchedule = form.Activex1.nativeObject
var_Calendar = oSchedule.Calendar
var_Calendar.FirstWeekDay = var_Calendar.LocFirstWeekDay
var_Calendar.MonthNames = var_Calendar.LocMonthNames
var_Calendar.WeekDays = var_Calendar.LocWeekDays
var_Calendar.AMPM = var_Calendar.LocAMPM
```

XBasic (Alpha Five)

Dim oSchedule as P Dim var_Calendar as P

```
oSchedule = topparent:CONTROL_ACTIVEX1.activex
var_Calendar = oSchedule.Calendar
var_Calendar.FirstWeekDay = var_Calendar.LocFirstWeekDay
var_Calendar.MonthNames = var_Calendar.LocMonthNames
var_Calendar.WeekDays = var_Calendar.LocWeekDays
var_Calendar.AMPM = var_Calendar.LocAMPM
```

Visual Objects

local var_Calendar as ICalendar

var_Calendar := oDCOCX_Exontrol1:Calendar
var_Calendar:FirstWeekDay := var_Calendar:LocFirstWeekDay
var_Calendar:MonthNames := var_Calendar:LocMonthNames
var_Calendar:WeekDays := var_Calendar:LocWeekDays
var_Calendar:AMPM := var_Calendar:LocAMPM

PowerBuilder

OleObject oSchedule,var_Calendar oSchedule = ole_1.Object var_Calendar = oSchedule.Calendar var_Calendar.FirstWeekDay = var_Calendar.LocFirstWeekDay var_Calendar.MonthNames = var_Calendar.LocMonthNames var_Calendar.WeekDays = var_Calendar.LocWeekDays var_Calendar.MPM = var_Calendar.LocMPM

property Calendar.LocWeekDays as String

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

Туре	Description
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 gets the locale list of names for each week day as indicated by current regional settings. The <%d1%>, <%d2%>, <%d3%>, <%ddd%> or <%dddd%> HTML tags indicates the week day, as appropriate set by the WeekDays property. The <%loc_d1%>, <%loc_d2%>, <%loc_d3%>, <%loc_ddd%> or <%loc_dddd%> HTML tags indicates the week day, as appropriate set by the WeekDays property, using the current user regional and language settings (LocAMPM property). The LocFirstWeekDay property indicates the first day of the week, using the current user regional and language settings. The LocAMPM property specifies the list of name of the months, using the current user regional and language settings. The LocAMPM property specifies the AM/PM time indicators, using the current user regional and language settings.

The following samples set the current view to display the locate date/time as set in the current regional settings.

VBA (MS Access, Excell...)

```
With Schedule1
With .Calendar
.FirstWeekDay = .LocFirstWeekDay
.MonthNames = .LocMonthNames
.WeekDays = .LocWeekDays
.AMPM = .LocAMPM
End With
```

VB6

With Schedule1 With .Calendar .FirstWeekDay = .**LocFirstWeekDay** .MonthNames = .**LocMonthNames**

```
.WeekDays = .LocWeekDays
.AMPM = .LocAMPM
End With
End With
```

VB.NET

With Exschedule1 With .Calendar .FirstWeekDay = .**LocFirstWeekDay** .MonthNames = .**LocMonthNames** .WeekDays = .**LocWeekDays** .AMPM = .**LocAMPM** End With End With

VB.NET for /COM

With AxSchedule1 With .Calendar .FirstWeekDay = .**LocFirstWeekDay** .MonthNames = .**LocMonthNames** .WeekDays = .**LocWeekDays** .AMPM = .**LocAMPM** End With End With

C++

```
/*
  Copy and paste the following directives to your header file as
    it defines the namespace 'EXSCHEDULELib' for the library: 'ExSchedule 1.0 Control
Library'
  #import <ExSchedule.dll>
    using namespace EXSCHEDULELib;
*/
EXSCHEDULELib::ISchedulePtr spSchedule1 = GetDlgItem(IDC_SCHEDULE1)-
```

```
>GetControlUnknown();
EXSCHEDULELib::ICalendarPtr var_Calendar = spSchedule1->GetCalendar();
var_Calendar->PutFirstWeekDay(var_Calendar->GetLocFirstWeekDay());
var_Calendar->PutMonthNames(var_Calendar->GetLocMonthNames());
var_Calendar->PutWeekDays(var_Calendar->GetLocWeekDays());
var_Calendar->PutAMPM(var_Calendar->GetLocAMPM());
```

C++ Builder

Exschedulelib_tlb::ICalendarPtr var_Calendar = Schedule1->Calendar; var_Calendar->FirstWeekDay = var_Calendar->LocFirstWeekDay; var_Calendar->MonthNames = var_Calendar->LocMonthNames; var_Calendar->WeekDays = var_Calendar->LocWeekDays; var_Calendar->AMPM = var_Calendar->LocAMPM;

C#

exontrol.EXSCHEDULELib.Calendar var_Calendar = exschedule1.Calendar; var_Calendar.FirstWeekDay = var_Calendar.LocFirstWeekDay; var_Calendar.MonthNames = var_Calendar.LocMonthNames; var_Calendar.WeekDays = var_Calendar.LocWeekDays; var_Calendar.AMPM = var_Calendar.LocAMPM;

JavaScript

```
<OBJECT classid="clsid:9B09E13D-7A88-4299-9DBE-383380435377"
id="Schedule1"></OBJECT>
<SCRIPT LANGUAGE="JScript">
var var_Calendar = Schedule1.Calendar;
var_Calendar.FirstWeekDay = var_Calendar.LocFirstWeekDay;
var_Calendar.MonthNames = var_Calendar.LocMonthNames;
var_Calendar.WeekDays = var_Calendar.LocWeekDays;
var_Calendar.WeekDays = var_Calendar.LocWeekDays;
var_Calendar.AMPM = var_Calendar.LocAMPM;
</SCRIPT>
```

EXSCHEDULELib.Calendar var_Calendar = axSchedule1.Calendar; var_Calendar.FirstWeekDay = var_Calendar.LocFirstWeekDay; var_Calendar.MonthNames = var_Calendar.LocMonthNames; var_Calendar.WeekDays = var_Calendar.LocWeekDays; var_Calendar.AMPM = var_Calendar.LocAMPM;

X++ (Dynamics Ax 2009)

```
public void init()
{
    COM com_Calendar;
    anytype var_Calendar;
    ;
    super();
    var_Calendar = exschedule1.Calendar(); com_Calendar = var_Calendar;
        com_Calendar.FirstWeekDay(com_Calendar.LocFirstWeekDay());
        com_Calendar.MonthNames(com_Calendar.LocMonthNames());
        com_Calendar.WeekDays(com_Calendar.LocWeekDays());
        com_Calendar.WeekDays(com_Calendar.LocWeekDays());
        com_Calendar.AMPM(com_Calendar.LocAMPM());
    }
}
```

Delphi 8 (.NET only)

```
with AxSchedule1 do
begin
with Calendar do
begin
FirstWeekDay := LocFirstWeekDay;
MonthNames := LocMonthNames;
WeekDays := LocWeekDays;
AMPM := LocAMPM;
end;
end
```

Delphi (standard)

```
with Schedule1 do
begin
with Calendar do
begin
FirstWeekDay := LocFirstWeekDay;
MonthNames := LocMonthNames;
WeekDays := LocWeekDays;
AMPM := LocAMPM;
end;
end
```

VFP

```
with thisform.Schedule1
with .Calendar
.FirstWeekDay = .LocFirstWeekDay
.MonthNames = .LocMonthNames
.WeekDays = .LocWeekDays
.AMPM = .LocAMPM
endwith
endwith
```

dBASE Plus

local oSchedule,var_Calendar

```
oSchedule = form.Activex1.nativeObject
var_Calendar = oSchedule.Calendar
var_Calendar.FirstWeekDay = var_Calendar.LocFirstWeekDay
var_Calendar.MonthNames = var_Calendar.LocMonthNames
var_Calendar.WeekDays = var_Calendar.LocWeekDays
var_Calendar.AMPM = var_Calendar.LocAMPM
```

XBasic (Alpha Five)

Dim oSchedule as P

```
oSchedule = topparent:CONTROL_ACTIVEX1.activex
var_Calendar = oSchedule.Calendar
var_Calendar.FirstWeekDay = var_Calendar.LocFirstWeekDay
var_Calendar.MonthNames = var_Calendar.LocMonthNames
var_Calendar.WeekDays = var_Calendar.LocWeekDays
var_Calendar.AMPM = var_Calendar.LocAMPM
```

Visual Objects

local var_Calendar as ICalendar

Dim var Calendar as P

var_Calendar := oDCOCX_Exontrol1:Calendar
var_Calendar:FirstWeekDay := var_Calendar:LocFirstWeekDay
var_Calendar:MonthNames := var_Calendar:LocMonthNames
var_Calendar:WeekDays := var_Calendar:LocWeekDays
var_Calendar:AMPM := var_Calendar:LocAMPM

PowerBuilder

OleObject oSchedule,var_Calendar

oSchedule = ole_1.Object var_Calendar = oSchedule.Calendar var_Calendar.FirstWeekDay = var_Calendar.LocFirstWeekDay var_Calendar.MonthNames = var_Calendar.LocMonthNames var_Calendar.WeekDays = var_Calendar.LocWeekDays var_Calendar.AMPM = var_Calendar.LocAMPM

property Calendar.LongDateFormat as String

Indicates the long date format.

Туре	Description
String	A String expression that defines the long date format. The LongDateFormat suports tags as described bellow.

By default, the LongDateFormat property is "<%loc_ldate%>", so the format of the long date as defined in the regional settings is currently used. The

KnownProperty(exEventDisplayLongMargins) property uses the LongDateFormat property to display the event's margins in a long date format. For instance, an all-day event (<u>AllDayEvent</u> property) displays the starting and ending margins of the event in a long date format. The <u>ShortDateFormat</u> property defines the short date format to be used when label properties includes the <%=%256%> TAG.

In conclusion, the LongDateFormat property defines the long date format being used to display the event's margins, when the <%**=%257**%> is included in the *label* properties such as:

- <u>DefaultEventLongLabel</u>, defines the HTML labels for events, when it fit entirely in the event's body.
- <u>DefaultEventShortLabel</u>, defines the labels for events (no HTML attribute is applied), when it does not fit the event's body
- CreateEventLabel, defines the label when creating a new event by dragging
- UpdateEventsLabel, defines the label of the events being moved or resized at runtime
- Event.<u>ShortLabel</u>, defines the event's short label, or the label to be shown when the LongLabel does not fit entirely the event's body.
- Event.<u>LongLabel</u>, defines the event's HTML long label, when it fits the body. If the LongLabel does not fit entirely the event's body, the ShortLabel is displayed instead.
- Event.<u>ExtraLabel</u>, defines the event's extra HTML label. The event's ExtraLabel is displayed ONLY, if the LongLabel fits the event's body

The property supports the following TAGs:

- <%d%> Day of the month in one or two numeric digits, as needed (1 to 31).
- <%dd%> Day of the month in two numeric digits (01 to 31).
- <%d1%> First letter of the weekday (S to S). (Use the <u>WeekDays</u> property to specify the name of the days in the week)
- <%d2%> First two letters of the weekday (Su to Sa). (Use the <u>WeekDays</u> property to specify the name of the days in the week)
- <%d3%> First three letters of the weekday (Sun to Sat). (Use the <u>WeekDays</u> property to specify the name of the days in the week)

- <%ddd%> First three letters of the weekday (Sun to Sat). (Use the <u>WeekDays</u> property to specify the name of the days in the week). You can use the <%loc_ddd%> that indicates the day of week as a three-letter abbreviation using the current user regional and language settings.
- <%loc_ddd%> Indicates the day of week as a three-letter abbreviation using the current user regional and language settings.
- <%dddd%> Full name of the weekday (Sunday to Saturday). (Use the <u>WeekDays</u> property to specify the name of the days in the week). You can use the <%loc_dddd%> that indicates day of week as its full name using the current user regional and language settings.
- <%loc_dddd%> Indicates day of week as its full name using the current user regional and language settings.
- <%i%> Displays the number instead the date. For instance, you can display numbers as 1000, 1001, 1002, 1003, instead dates. (the valid range is from -647,434 to 2,958,465)
- <%**w**%> Day of the week (1 to 7).
- <%**ww**%> Week of the year (1 to 53).
- <%m%> Month of the year in one or two numeric digits, as needed (1 to 12).
- <%mr%> Month of the year in Roman numerals, as needed (I to XII).
- <%mm%> Month of the year in two numeric digits (01 to 12).
- <%m1%> First letter of the month (J to D). (Use the <u>MonthNames</u> property to specify the name of the months in the year)
- <%m2%> First two letters of the month (Ja to De). (Use the <u>MonthNames</u> property to specify the name of the months in the year)
- <%m3%> First three letters of the month (Jan to Dec). (Use the <u>MonthNames</u> property to specify the name of the months in the year)
- <%mmm%> First three letters of the month (Jan to Dec). (Use the MonthNames property to specify the name of the months in the year). You can use the <%loc_mmm%> that indicates month as a three-letter abbreviation using the current user regional and language settings.
- <%loc_mmm%> Indicates month as a three-letter abbreviation using the current user regional and language settings.
- <%mmm%> Full name of the month (January to December). (Use the MonthNames property to specify the name of the months in the year). You can use the <%loc_mmm%> that indicates month as its full name using the current user regional and language settings.
- <%loc_mmmm%> Indicates month as its full name using the current user regional and language settings.
- <%q%> Date displayed as the quarter of the year (1 to 4).
- $\langle \mathbf{y} \mathbf{y} \rangle$ Number of the day of the year (1 to 366).
- <%yy%> Last two digits of the year (01 to 99).
- <%**yyyy**%> Full year (0100 to 9999).

- <%hy%> Date displayed as the half of the year (1 to 2).
- <%loc_gg%> Indicates period/era using the current user regional and language settings.
- <%loc_sdate%> Indicates the date in the short format using the current user regional and language settings.
- <%loc_ldate%> Indicates the date in the long format using the current user regional and language settings.
- <%loc_dsep%> Indicates the date separator using the current user regional and language settings (/).
- <%h%> Hour in one or two digits, as needed (0 to 23).
- <%**hh**%> Hour in two digits (00 to 23).
- <%n%> Minute in one or two digits, as needed (0 to 59).
- <%**nn**%> Minute in two digits (00 to 59).
- <%**s**%> Second in one or two digits, as needed (0 to 59).
- <%**ss**%> Second in two digits (00 to 59).
- <%AM/PM%> Twelve-hour clock with the uppercase letters "AM" or "PM", as appropriate. (Use the <u>AMPM</u> property to specify the name of the AM and PM indicators). You can use the <%loc_AM/PM%> that indicates the time marker such as AM or PM using the current user regional and language settings. You can use <%loc_A/P%> that indicates the one character time marker such as A or P using the current user regional and language settings.
- <%loc_AM/PM%> Indicates the time marker such as AM or PM using the current user regional and language settings.
- <%loc_A/P%> Indicates the one character time marker such as A or P using the current user regional and language settings.
- <%loc_time%> Indicates the time using the current user regional and language settings.
- <%loc_time24%> Indicates the time in 24 hours format without a time marker using the current user regional and language settings.
- <%loc_tsep%> indicates the time separator using the current user regional and language settings (:).

The following tags are displayed based on the user's Regional and Language Options:

- <%loc_sdate%> Indicates the date in the short format using the current user settings.
- <%loc_ldate%> Indicates the date in the long format using the current user settings.
- <%loc_ddd%> Indicates day of week as a three-letter abbreviation using the current user settings.
- <%loc_dddd%> Indicates day of week as its full name using the current user settings.
- <%loc_mmm%> Indicates month as a three-letter abbreviation using the current user

settings.

- <%loc_mmmm%> Indicates month as its full name using the current user settings.
- <%loc_gg%> Indicates period/era using the current user settings.
- <%loc_dsep%> Indicates the date separator using the current user settings.
- <%loc_time%> Indicates the time using the current user settings.
- <%loc_time24%> Indicates the time in 24 hours format without a time marker using the current user settings.
- <%loc_AM/PM%> Indicates the time marker such as AM or PM using the current user settings.
- <%loc_A/P%> Indicates the one character time marker such as A or P using the current user settings.
- <%loc_tsep%> Indicates the time separator using the current user settings.
- <%loc_y%> Represents the Year only by the last digit, using current regional settings.
- <%loc_yy%> Represents the Year only by the last two digits, using current regional settings. A leading zero is added for single-digit years.
- <%loc_yyyy%> Represents the Year by a full four or five digits, depending on the calendar used. Thai Buddhist and Korean calendars have five-digit years. The "yyyy" pattern shows five digits for these two calendars, and four digits for all other supported calendars. Calendars that have single-digit or two-digit years, such as for the Japanese Emperor era, are represented differently. A single-digit year is represented with a leading zero, for example, "03". A two-digit year is represented with two digits, for example, "13". No additional leading zeros are displayed.

property Calendar.LongTimeFormat as String

Indicates the long time format.

Туре	Description
String	A String expression that defines the long time format. The LongTimeFormat supports tags as described bellow.

By default, the LongTimeFormat property is "<%hh%>:<%nn%>:<%ss%> <%AM/PM%>", as 08:15:00 AM. The KnownProperty(exEventDisplayLongMargins) property uses the LongTimeFormat property to display the event's margins in a long time format. The LongTimeFormat property is used by label properties if <%=%257%> TAG is included as explained bellow. The ShortTimeFormat property defines the short time format to be used when label properties includes the <%=%256%> TAG.

In conclusion, the LongTimeFormat property defines the short time format being used to display the event's margins, when the <%**=%257**%> is included in the *label* properties such as:

- <u>DefaultEventLongLabel</u>, defines the HTML labels for events, when it fit entirely in the event's body.
- <u>DefaultEventShortLabel</u>, defines the labels for events (no HTML attribute is applied), when it does not fit the event's body
- CreateEventLabel, defines the label when creating a new event by dragging
- UpdateEventsLabel, defines the label of the events being moved or resized at runtime
- Event.<u>ShortLabel</u>, defines the event's short label, or the label to be shown when the LongLabel does not fit entirely the event's body. The ShortLabel displays no HTML tags
- Event.<u>LongLabel</u>, defines the event's HTML long label, when it fits the body. If the LongLabel does not fit entirely the event's body, the ShortLabel is displayed instead.
- Event.<u>ExtraLabel</u>, defines the event's extra HTML label. The event's ExtraLabel is displayed ONLY, if the LongLabel fits the event's body

The property supports the following TAGs:

- <%d%> Day of the month in one or two numeric digits, as needed (1 to 31).
- <%**dd**%> Day of the month in two numeric digits (01 to 31).
- <%d1%> First letter of the weekday (S to S). (Use the <u>WeekDays</u> property to specify the name of the days in the week)
- <%d2%> First two letters of the weekday (Su to Sa). (Use the <u>WeekDays</u> property to specify the name of the days in the week)
- <%d3%> First three letters of the weekday (Sun to Sat). (Use the <u>WeekDays</u> property to specify the name of the days in the week)
- <%ddd%> First three letters of the weekday (Sun to Sat). (Use the WeekDays

property to specify the name of the days in the week). You can use the <%loc_ddd%> that indicates the day of week as a three-letter abbreviation using the current user regional and language settings.

- <%loc_ddd%> Indicates the day of week as a three-letter abbreviation using the current user regional and language settings.
- <%dddd%> Full name of the weekday (Sunday to Saturday). (Use the <u>WeekDays</u> property to specify the name of the days in the week). You can use the <%loc_dddd%> that indicates day of week as its full name using the current user regional and language settings.
- <%loc_dddd%> Indicates day of week as its full name using the current user regional and language settings.
- <%i%> Displays the number instead the date. For instance, you can display numbers as 1000, 1001, 1002, 1003, instead dates. (the valid range is from -647,434 to 2,958,465)
- <%**w**%> Day of the week (1 to 7).
- <%ww%> Week of the year (1 to 53).
- <%m%> Month of the year in one or two numeric digits, as needed (1 to 12).
- <%mr%> Month of the year in Roman numerals, as needed (I to XII).
- <%mm%> Month of the year in two numeric digits (01 to 12).
- <%m1%> First letter of the month (J to D). (Use the <u>MonthNames</u> property to specify the name of the months in the year)
- <%m2%> First two letters of the month (Ja to De). (Use the <u>MonthNames</u> property to specify the name of the months in the year)
- <%m3%> First three letters of the month (Jan to Dec). (Use the <u>MonthNames</u> property to specify the name of the months in the year)
- <%mmm%> First three letters of the month (Jan to Dec). (Use the <u>MonthNames</u> property to specify the name of the months in the year). You can use the <%loc_mmm%> that indicates month as a three-letter abbreviation using the current user regional and language settings.
- <%loc_mmm%> Indicates month as a three-letter abbreviation using the current user regional and language settings.
- <%mmm%> Full name of the month (January to December). (Use the MonthNames property to specify the name of the months in the year). You can use the <%loc_mmm%> that indicates month as its full name using the current user regional and language settings.
- <%loc_mmmm%> Indicates month as its full name using the current user regional and language settings.
- <%q%> Date displayed as the quarter of the year (1 to 4).
- $\langle \mathbf{y} \rangle$ Number of the day of the year (1 to 366).
- <%**yy**%> Last two digits of the year (01 to 99).
- <%**yyyy**%> Full year (0100 to 9999).
- <%hy%> Date displayed as the half of the year (1 to 2).

- <%loc_gg%> Indicates period/era using the current user regional and language settings.
- <%loc_sdate%> Indicates the date in the short format using the current user regional and language settings.
- <%loc_ldate%> Indicates the date in the long format using the current user regional and language settings.
- <%loc_dsep%> Indicates the date separator using the current user regional and language settings (/).
- <%h%> Hour in one or two digits, as needed (0 to 23).
- <%**hh**%> Hour in two digits (00 to 23).
- <%**n**%> Minute in one or two digits, as needed (0 to 59).
- <%nn%> Minute in two digits (00 to 59).
- <%s%> Second in one or two digits, as needed (0 to 59).
- <%**ss**%> Second in two digits (00 to 59).
- <%AM/PM%> Twelve-hour clock with the uppercase letters "AM" or "PM", as appropriate. (Use the <u>AMPM</u> property to specify the name of the AM and PM indicators). You can use the <%loc_AM/PM%> that indicates the time marker such as AM or PM using the current user regional and language settings. You can use <%loc_A/P%> that indicates the one character time marker such as A or P using the current user regional and language settings.
- <%loc_AM/PM%> Indicates the time marker such as AM or PM using the current user regional and language settings.
- <%loc_A/P%> Indicates the one character time marker such as A or P using the current user regional and language settings.
- <%loc_time%> Indicates the time using the current user regional and language settings.
- <%loc_time24%> Indicates the time in 24 hours format without a time marker using the current user regional and language settings.
- <%loc_tsep%> indicates the time separator using the current user regional and language settings (:).

The following tags are displayed based on the user's Regional and Language Options:

- <%loc_sdate%> Indicates the date in the short format using the current user settings.
- <%loc_ldate%> Indicates the date in the long format using the current user settings.
- <%loc_ddd%> Indicates day of week as a three-letter abbreviation using the current user settings.
- <%loc_dddd%> Indicates day of week as its full name using the current user settings.
- <%loc_mmm%> Indicates month as a three-letter abbreviation using the current user settings.

- <%loc_mmmm%> Indicates month as its full name using the current user settings.
- <%loc_gg%> Indicates period/era using the current user settings.
- <%loc_dsep%> Indicates the date separator using the current user settings.
- <%loc_time%> Indicates the time using the current user settings.
- <%loc_time24%> Indicates the time in 24 hours format without a time marker using the current user settings.
- <%loc_AM/PM%> Indicates the time marker such as AM or PM using the current user settings.
- <%loc_A/P%> Indicates the one character time marker such as A or P using the current user settings.
- <%loc_tsep%> Indicates the time separator using the current user settings.
- <%loc_y%> Represents the Year only by the last digit, using current regional settings.
- <%loc_yy%> Represents the Year only by the last two digits, using current regional settings. A leading zero is added for single-digit years.
- <%loc_yyyy%> Represents the Year by a full four or five digits, depending on the calendar used. Thai Buddhist and Korean calendars have five-digit years. The "yyy" pattern shows five digits for these two calendars, and four digits for all other supported calendars. Calendars that have single-digit or two-digit years, such as for the Japanese Emperor era, are represented differently. A single-digit year is represented with a leading zero, for example, "03". A two-digit year is represented with two digits, for example, "13". No additional leading zeros are displayed.

property Calendar.MaxDate as Date

Retrieves or sets the max date.

Туре	Description
Date	A DATE expression that specifies the upper margin that the calendar can browse.

By default, the MaxDate is 12/31/9999. You can use the MaxDate property to limit the dates that the calendar can show. When the calendar has no other dates to show, the left or right arrows are not shown. The <u>MinDate</u> property indicates the lower margin that the calendar can show. The <u>Background</u>(exCalendarArrowLeft) and Background(exCalendarArrowRight) indicates the visual aspect of the left and right arrows in the calendar panel. The <u>MinDate</u> property of the Event can be used to specify the lower limit of the event when it is resized or moved. The <u>MaxDate</u> property of the Event can be used to specify the upper limit of the event when it is resized or moved. The <u>DisableZoneFormat</u> property may be used to specify the dates to be shown as disabled.

The samples shows how you can limit the schedule view to a single month.

VBA (MS Access, Excell...)

```
With Schedule1

.ScrollBars = 0

.AllowMoveSchedule = 0

With .Calendar

.Selection = #1/10/2001#

.MinDate = #1/1/2001#

.MaxDate = #1/31/2001#

End With

End With
```

VB6

```
With Schedule1

.ScrollBars = exNoScroll

.AllowMoveSchedule = exDisallow

With .Calendar

.Selection = #1/10/2001#

.MinDate = #1/1/2001#

.MaxDate = #1/31/2001#
```

```
End With
End With
```

VB.NET

```
With Exschedule1

.ScrollBars = exontrol.EXSCHEDULELib.ScrollBarsEnum.exNoScroll

.AllowMoveSchedule = exontrol.EXSCHEDULELib.AllowKeysEnum.exDisallow

With .Calendar

.Selection = #1/10/2001#

.MinDate = #1/1/2001#

.MaxDate = #1/31/2001#

End With

End With
```

VB.NET for /COM

```
With AxSchedule1

.ScrollBars = EXSCHEDULELib.ScrollBarsEnum.exNoScroll

.AllowMoveSchedule = EXSCHEDULELib.AllowKeysEnum.exDisallow

With .Calendar

.Selection = #1/10/2001#

.MinDate = #1/1/2001#

.MaxDate = #1/31/2001#

End With

End With
```

C++

```
/*
  Copy and paste the following directives to your header file as
    it defines the namespace 'EXSCHEDULELib' for the library: 'ExSchedule 1.0 Control
Library'
  #import <ExSchedule.dll>
    using namespace EXSCHEDULELib;
*/
EXSCHEDULELib::ISchedulePtr spSchedule1 = GetDlgItem(IDC_SCHEDULE1)-
```

```
>GetControlUnknown();
spSchedule1->PutScrollBars(EXSCHEDULELib::exNoScroll);
spSchedule1->PutAllowMoveSchedule(EXSCHEDULELib::exDisallow);
EXSCHEDULELib::ICalendarPtr var_Calendar = spSchedule1->GetCalendar();
var_Calendar->PutSelection("1/10/2001");
var_Calendar->PutMinDate("1/1/2001");
var_Calendar->PutMaxDate("1/31/2001");
```

C++ Builder

```
Schedule1->ScrollBars = Exschedulelib_tlb::ScrollBarsEnum::exNoScroll;
Schedule1->AllowMoveSchedule = Exschedulelib_tlb::AllowKeysEnum::exDisallow;
Exschedulelib_tlb::ICalendarPtr var_Calendar = Schedule1->Calendar;
var_Calendar->set_Selection(TVariant(TDateTime(2001,1,10).operator double()));
var_Calendar->MinDate = TDateTime(2001,1,1).operator double();
var_Calendar->MaxDate = TDateTime(2001,1,31).operator double();
```

C#

```
exschedule1.ScrollBars = exontrol.EXSCHEDULELib.ScrollBarsEnum.exNoScroll;
exschedule1.AllowMoveSchedule =
exontrol.EXSCHEDULELib.AllowKeysEnum.exDisallow;
exontrol.EXSCHEDULELib.Calendar var_Calendar = exschedule1.Calendar;
var_Calendar.Selection =
Convert.ToDateTime("1/10/2001",System.Globalization.CultureInfo.GetCultureInfo("en-
US"));
var_Calendar.MinDate =
Convert.ToDateTime("1/1/2001",System.Globalization.CultureInfo.GetCultureInfo("en-
US"));
var_Calendar.MaxDate =
Convert.ToDateTime("1/31/2001",System.Globalization.CultureInfo.GetCultureInfo("en-
US"));
```

```
<OBJECT classid="clsid:9B09E13D-7A88-4299-9DBE-383380435377"
id="Schedule1"> </OBJECT>
<SCRIPT LANGUAGE="JScript">
Schedule1.ScrollBars = 0;
Schedule1.AllowMoveSchedule = 0;
var var_Calendar = Schedule1.Calendar;
var_Calendar.Selection = "1/10/2001";
var_Calendar.MinDate = "1/1/2001";
var_Calendar.MaxDate = "1/31/2001";
```

C# for /COM

```
axSchedule1.ScrollBars = EXSCHEDULELib.ScrollBarsEnum.exNoScroll;
axSchedule1.AllowMoveSchedule = EXSCHEDULELib.AllowKeysEnum.exDisallow;
EXSCHEDULELib.Calendar var_Calendar = axSchedule1.Calendar;
var_Calendar.Selection =
Convert.ToDateTime("1/10/2001",System.Globalization.CultureInfo.GetCultureInfo("en-
```

```
US"));
var_Calendar.MinDate =
```

```
Convert.ToDateTime("1/1/2001",System.Globalization.CultureInfo.GetCultureInfo("en-US"));
```

```
var_Calendar.MaxDate =
```

```
Convert.ToDateTime("1/31/2001",System.Globalization.CultureInfo.GetCultureInfo("en-US"));
```

X++ (Dynamics Ax 2009)

```
public void init()
{
    COM com_Calendar;
    anytype var_Calendar;
    ;
    super();
```

```
exschedule1.ScrollBars(0/*exNoScroll*/);
exschedule1.AllowMoveSchedule(0/*exDisallow*/);
var_Calendar = exschedule1.Calendar(); com_Calendar = var_Calendar;
com_Calendar.Selection(COMVariant::createFromDate(str2Date("1/10/2001",213)));
com_Calendar.MinDate(str2Date("1/1/2001",213));
com_Calendar.MaxDate(str2Date("1/31/2001",213));
}
```

Delphi 8 (.NET only)

```
with AxSchedule1 do
begin
   ScrollBars := EXSCHEDULELib.ScrollBarsEnum.exNoScroll;
   AllowMoveSchedule := EXSCHEDULELib.AllowKeysEnum.exDisallow;
   with Calendar do
   begin
      Selection := '1/10/2001';
      MinDate := '1/1/2001';
      MaxDate := '1/31/2001';
   end;
end
```

Delphi (standard)

```
with Schedule1 do
begin
   ScrollBars := EXSCHEDULELib_TLB.exNoScroll;
   AllowMoveSchedule := EXSCHEDULELib_TLB.exDisallow;
   with Calendar do
   begin
     Selection := '1/10/2001';
   MinDate := '1/1/2001';
   MaxDate := '1/31/2001';
   end;
end
```

```
with thisform.Schedule1
.ScrollBars = 0
.AllowMoveSchedule = 0
with .Calendar
.Selection = {^2001-1-10}
.MinDate = {^2001-1-1}
.MaxDate = {^2001-1-31}
endwith
endwith
```

dBASE Plus

local oSchedule,var_Calendar

```
oSchedule = form.Activex1.nativeObject
oSchedule.ScrollBars = 0
oSchedule.AllowMoveSchedule = 0
var_Calendar = oSchedule.Calendar
var_Calendar.Selection = "01/10/2001"
var_Calendar.MinDate = "01/01/2001"
var_Calendar.MaxDate = "01/31/2001"
```

XBasic (Alpha Five)

```
Dim oSchedule as P
Dim var_Calendar as P
oSchedule = topparent:CONTROL_ACTIVEX1.activex
oSchedule.ScrollBars = 0
oSchedule.AllowMoveSchedule = 0
var_Calendar = oSchedule.Calendar
var_Calendar.Selection = {01/10/2001}
var_Calendar.MinDate = {01/01/2001}
var_Calendar.MaxDate = {01/31/2001}
```

Visual Objects

local var_Calendar as ICalendar

oDCOCX_Exontrol1:**ScrollBars** := exNoScroll oDCOCX_Exontrol1:AllowMoveSchedule := exDisallow var_Calendar := oDCOCX_Exontrol1:Calendar var_Calendar:Selection := SToD("20010110") var_Calendar:**MinDate** := SToD("20010101") var_Calendar:**MaxDate** := SToD("20010131")

PowerBuilder

OleObject oSchedule,var_Calendar

```
oSchedule = ole_1.Object
oSchedule.ScrollBars = 0
oSchedule.AllowMoveSchedule = 0
var_Calendar = oSchedule.Calendar
var_Calendar.Selection = 2001-01-10
var_Calendar.MinDate = 2001-01-01
var_Calendar.MaxDate = 2001-01-31
```

property Calendar.MaxMonthX as Long

Specifies the maximum number of months horizontally displayed.

Туре	Description
Long	A long expression that specifies the number of months that the calendar panel may display horizontally.

By default, the MaxMonthX property is 1, which indicates that a single month is being displayed horizontally. You can use the <u>MinMonthX</u>/MaxMonthX, <u>MinMonthY</u>/<u>MaxMonthY</u> to specify the months to be displayed on the calendar panel. The <u>OnResizeControl</u> property on OnResizeControlEnum.exHideSplitter Or OnResizeControlEnum.exChangePanels hides the calendar panel.

property Calendar.MaxMonthY as Long

Specifies the maximum number of months vertically displayed.

Туре	Description
Long	A long expression that specifies the number of months that the calendar panel may display vertically.

By default, the MaxMonthX property is 1, which indicates that a single month is being displayed horizontally. You can use the <u>MinMonthX/MaxMonthX</u>, <u>MinMonthY</u>/MaxMonthY to specify the months to be displayed on the calendar panel. The <u>OnResizeControl</u> property on OnResizeControlEnum.exHideSplitter Or OnResizeControlEnum.exChangePanels hides the calendar panel.

property Calendar.MinDate as Date

Retrieves or sets the min date.

Туре	Description
Date	A DATE expression that specifies the upper margin that the calendar can browse.

By default, the MinDate is 1/1/100. You can use the MinDate property to limit the dates that the calendar can show. When the calendar has no other dates to show, the left or right arrows are not shown. The MaxDate property indicates the upper margin that the calendar can show. The Background(exCalendarArrowLeft) and Background(exCalendarArrowRight) indicates the visual aspect of the left and right arrows in the calendar panel. The MinDate property of the Event can be used to specify the lower limit of the event when it is resized or moved. The MaxDate property of the Event can be used to specify the upper limit of the upper limit of the event when it is resized or moved. The DisableZoneFormat property may be used to specify the dates to be shown as disabled.

The samples shows how you can limit the schedule view to a single month.

VBA (MS Access, Excell...)

```
With Schedule1

.ScrollBars = 0

.AllowMoveSchedule = 0

With .Calendar

.Selection = #1/10/2001#

.MinDate = #1/1/2001#

.MaxDate = #1/31/2001#

End With

End With
```

VB6

```
With Schedule1

.ScrollBars = exNoScroll

.AllowMoveSchedule = exDisallow

With .Calendar

.Selection = #1/10/2001#

.MinDate = #1/1/2001#

.MaxDate = #1/31/2001#
```

```
End With
End With
```

VB.NET

```
With Exschedule1

.ScrollBars = exontrol.EXSCHEDULELib.ScrollBarsEnum.exNoScroll

.AllowMoveSchedule = exontrol.EXSCHEDULELib.AllowKeysEnum.exDisallow

With .Calendar

.Selection = #1/10/2001#

.MinDate = #1/1/2001#

.MaxDate = #1/31/2001#

End With

End With
```

VB.NET for /COM

```
With AxSchedule1

.ScrollBars = EXSCHEDULELib.ScrollBarsEnum.exNoScroll

.AllowMoveSchedule = EXSCHEDULELib.AllowKeysEnum.exDisallow

With .Calendar

.Selection = #1/10/2001#

.MinDate = #1/1/2001#

.MaxDate = #1/31/2001#

End With

End With
```

C++

```
/*
  Copy and paste the following directives to your header file as
    it defines the namespace 'EXSCHEDULELib' for the library: 'ExSchedule 1.0 Control
Library'
  #import <ExSchedule.dll>
    using namespace EXSCHEDULELib;
*/
EXSCHEDULELib::ISchedulePtr spSchedule1 = GetDlgItem(IDC_SCHEDULE1)-
```

```
>GetControlUnknown();
spSchedule1->PutScrollBars(EXSCHEDULELib::exNoScroll);
spSchedule1->PutAllowMoveSchedule(EXSCHEDULELib::exDisallow);
EXSCHEDULELib::ICalendarPtr var_Calendar = spSchedule1->GetCalendar();
var_Calendar->PutSelection("1/10/2001");
var_Calendar->PutMinDate("1/1/2001");
var_Calendar->PutMaxDate("1/31/2001");
```

C++ Builder

```
Schedule1->ScrollBars = Exschedulelib_tlb::ScrollBarsEnum::exNoScroll;
Schedule1->AllowMoveSchedule = Exschedulelib_tlb::AllowKeysEnum::exDisallow;
Exschedulelib_tlb::ICalendarPtr var_Calendar = Schedule1->Calendar;
var_Calendar->set_Selection(TVariant(TDateTime(2001,1,10).operator double()));
var_Calendar->MinDate = TDateTime(2001,1,1).operator double();
var_Calendar->MaxDate = TDateTime(2001,1,31).operator double();
```

C#

```
exschedule1.ScrollBars = exontrol.EXSCHEDULELib.ScrollBarsEnum.exNoScroll;
exschedule1.AllowMoveSchedule =
exontrol.EXSCHEDULELib.AllowKeysEnum.exDisallow;
exontrol.EXSCHEDULELib.Calendar var_Calendar = exschedule1.Calendar;
var_Calendar.Selection =
Convert.ToDateTime("1/10/2001",System.Globalization.CultureInfo.GetCultureInfo("en-
US"));
var_Calendar.MinDate =
Convert.ToDateTime("1/1/2001",System.Globalization.CultureInfo.GetCultureInfo("en-
US"));
var_Calendar.MaxDate =
Convert.ToDateTime("1/31/2001",System.Globalization.CultureInfo.GetCultureInfo("en-
US"));
```

```
<OBJECT classid="clsid:9B09E13D-7A88-4299-9DBE-383380435377"
id="Schedule1"> </OBJECT>
<SCRIPT LANGUAGE="JScript">
Schedule1.ScrollBars = 0;
Schedule1.AllowMoveSchedule = 0;
var var_Calendar = Schedule1.Calendar;
var_Calendar.Selection = "1/10/2001";
var_Calendar.MinDate = "1/1/2001";
var_Calendar.MaxDate = "1/31/2001";
```

C# for /COM

```
axSchedule1.ScrollBars = EXSCHEDULELib.ScrollBarsEnum.exNoScroll;
axSchedule1.AllowMoveSchedule = EXSCHEDULELib.AllowKeysEnum.exDisallow;
EXSCHEDULELib.Calendar var_Calendar = axSchedule1.Calendar;
var_Calendar.Selection =
Convert.ToDateTime("1/10/2001",System.Globalization.CultureInfo.GetCultureInfo("en-
```

```
US"));
var_Calendar.MinDate =
```

```
Convert.ToDateTime("1/1/2001",System.Globalization.CultureInfo.GetCultureInfo("en-US"));
```

```
var_Calendar.MaxDate =
```

```
Convert.ToDateTime("1/31/2001",System.Globalization.CultureInfo.GetCultureInfo("en-US"));
```

X++ (Dynamics Ax 2009)

```
public void init()
{
    COM com_Calendar;
    anytype var_Calendar;
    ;
    super();
```

```
exschedule1.ScrollBars(0/*exNoScroll*/);
exschedule1.AllowMoveSchedule(0/*exDisallow*/);
var_Calendar = exschedule1.Calendar(); com_Calendar = var_Calendar;
com_Calendar.Selection(COMVariant::createFromDate(str2Date("1/10/2001",213)));
com_Calendar.MinDate(str2Date("1/1/2001",213));
com_Calendar.MaxDate(str2Date("1/31/2001",213));
}
```

Delphi 8 (.NET only)

```
with AxSchedule1 do
begin
   ScrollBars := EXSCHEDULELib.ScrollBarsEnum.exNoScroll;
   AllowMoveSchedule := EXSCHEDULELib.AllowKeysEnum.exDisallow;
   with Calendar do
   begin
      Selection := '1/10/2001';
      MinDate := '1/1/2001';
      MaxDate := '1/31/2001';
   end;
end
```

Delphi (standard)

```
with Schedule1 do
begin
   ScrollBars := EXSCHEDULELib_TLB.exNoScroll;
   AllowMoveSchedule := EXSCHEDULELib_TLB.exDisallow;
   with Calendar do
   begin
     Selection := '1/10/2001';
   MinDate := '1/1/2001';
   MaxDate := '1/31/2001';
   end;
end
```

```
with thisform.Schedule1
.ScrollBars = 0
.AllowMoveSchedule = 0
with .Calendar
.Selection = {^2001-1-10}
.MinDate = {^2001-1-1}
.MaxDate = {^2001-1-31}
endwith
endwith
```

dBASE Plus

local oSchedule,var_Calendar

```
oSchedule = form.Activex1.nativeObject
oSchedule.ScrollBars = 0
oSchedule.AllowMoveSchedule = 0
var_Calendar = oSchedule.Calendar
var_Calendar.Selection = "01/10/2001"
var_Calendar.MinDate = "01/01/2001"
var_Calendar.MaxDate = "01/31/2001"
```

XBasic (Alpha Five)

```
Dim oSchedule as P
Dim var_Calendar as P
oSchedule = topparent:CONTROL_ACTIVEX1.activex
oSchedule.ScrollBars = 0
oSchedule.AllowMoveSchedule = 0
var_Calendar = oSchedule.Calendar
var_Calendar.Selection = {01/10/2001}
var_Calendar.MinDate = {01/01/2001}
var_Calendar.MaxDate = {01/31/2001}
```

Visual Objects

local var_Calendar as ICalendar

oDCOCX_Exontrol1:**ScrollBars** := exNoScroll oDCOCX_Exontrol1:AllowMoveSchedule := exDisallow var_Calendar := oDCOCX_Exontrol1:Calendar var_Calendar:Selection := SToD("20010110") var_Calendar:**MinDate** := SToD("20010101") var_Calendar:**MaxDate** := SToD("20010131")

PowerBuilder

OleObject oSchedule,var_Calendar

```
oSchedule = ole_1.Object
oSchedule.ScrollBars = 0
oSchedule.AllowMoveSchedule = 0
var_Calendar = oSchedule.Calendar
var_Calendar.Selection = 2001-01-10
var_Calendar.MinDate = 2001-01-01
var_Calendar.MaxDate = 2001-01-31
```

property Calendar.MinMonthX as Long

Specifies the minimum number of months horizontally displayed.

Туре	Description
Long	A long expression that specifies the number of months that the calendar panel may display horizontally.

By default, the MaxMonthX property is 1, which indicates that a single month is being displayed horizontally. You can use the MinMonthX/<u>MaxMonthX</u>, <u>MinMonthY/MaxMonthY</u> to specify the months to be displayed on the calendar panel. The <u>OnResizeControl</u> property on OnResizeControlEnum.exHideSplitter Or OnResizeControlEnum.exChangePanels hides the calendar panel.

property Calendar.MinMonthY as Long

Specifies the minimum number of months vertically displayed.

Туре	Description
Long	A long expression that specifies the number of months that the calendar panel may display vertically.

By default, the MaxMonthX property is 1, which indicates that a single month is being displayed horizontally. You can use the <u>MinMonthX/MaxMonthX</u>, MinMonthY/<u>MaxMonthY</u> to specify the months to be displayed on the calendar panel. The <u>OnResizeControl</u> property on OnResizeControlEnum.exHideSplitter Or OnResizeControlEnum.exChangePanels hides the calendar panel.
property Calendar.MonthNames as String

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

Туре	Description
String	A String expression that indicates the name of the months within the year, separated by space.

By default, the MonthNames property is "January February March April May June July August September October November December". Use the LocMonthNames property to get the name of the months as indicated by current regional settings. The <%m1%>, <%m2%>, <%m3%>, <%mmmm%> HTML tags indicate the name of the month, as appropriate set by the MonthNames property. The <%loc_m1%>, <%loc_m2%>, <%loc_m3%>, <%loc_mmm%> HTML tags indicate the month using the current user regional and language settings (LocMonthNames property). The FirstWeekDay property indicates the first day of the week. The AMPM property specifies specifies the AM and PM indicators. The WeekDays property specifies the name of the days in the week.

The following samples set the current view to display the locate date/time as set in the current regional settings.

VBA (MS Access, Excell...)

```
With Schedule1
With .Calendar
.FirstWeekDay = .LocFirstWeekDay
.MonthNames = .LocMonthNames
.WeekDays = .LocWeekDays
.AMPM = .LocAMPM
End With
```

VB6

```
With Schedule1
With .Calendar
.FirstWeekDay = .LocFirstWeekDay
.MonthNames = .LocMonthNames
.WeekDays = .LocWeekDays
.AMPM = .LocAMPM
End With
```

VB.NET

With Exschedule1 With .Calendar .FirstWeekDay = .**LocFirstWeekDay** .MonthNames = .**LocMonthNames** .WeekDays = .**LocWeekDays** .AMPM = .**LocAMPM** End With End With

VB.NET for /COM

With AxSchedule1 With .Calendar .FirstWeekDay = .**LocFirstWeekDay** .MonthNames = .**LocMonthNames** .WeekDays = .**LocWeekDays** .AMPM = .**LocAMPM** End With End With

C++

Copy and paste the following directives to your header file as it defines the namespace 'EXSCHEDULELib' for the library: 'ExSchedule 1.0 Control Library'

```
#import <ExSchedule.dll>
using namespace EXSCHEDULELib;
*/
EXSCHEDULELib::ISchedulePtr spSchedule1 = GetDlgItem(IDC_SCHEDULE1)-
>GetControlUnknown();
EXSCHEDULELib::ICalendarPtr var_Calendar = spSchedule1->GetCalendar();
var_Calendar->PutFirstWeekDay(var_Calendar->GetLocFirstWeekDay());
```

```
var_Calendar->PutMonthNames(var_Calendar->GetLocMonthNames());
var_Calendar->PutWeekDays(var_Calendar->GetLocWeekDays());
var_Calendar->PutAMPM(var_Calendar->GetLocAMPM());
```

C++ Builder

```
Exschedulelib_tlb::ICalendarPtr var_Calendar = Schedule1->Calendar;
var_Calendar->FirstWeekDay = var_Calendar->LocFirstWeekDay;
var_Calendar->MonthNames = var_Calendar->LocMonthNames;
var_Calendar->WeekDays = var_Calendar->LocWeekDays;
var_Calendar->AMPM = var_Calendar->LocAMPM;
```

C#

```
exontrol.EXSCHEDULELib.Calendar var_Calendar = exschedule1.Calendar;
var_Calendar.FirstWeekDay = var_Calendar.LocFirstWeekDay;
var_Calendar.MonthNames = var_Calendar.LocMonthNames;
var_Calendar.WeekDays = var_Calendar.LocWeekDays;
var_Calendar.AMPM = var_Calendar.LocAMPM;
```

JavaScript

```
<OBJECT classid="clsid:9B09E13D-7A88-4299-9DBE-383380435377"
id="Schedule1"></OBJECT>
<SCRIPT LANGUAGE="JScript">
var var_Calendar = Schedule1.Calendar;
var_Calendar.FirstWeekDay = var_Calendar.LocFirstWeekDay;
var_Calendar.MonthNames = var_Calendar.LocMonthNames;
var_Calendar.WeekDays = var_Calendar.LocWeekDays;
var_Calendar.AMPM = var_Calendar.LocAMPM;
</SCRIPT>
```

C# for /COM

EXSCHEDULELib.Calendar var_Calendar = axSchedule1.Calendar;

```
var_Calendar.FirstWeekDay = var_Calendar.LocFirstWeekDay;
var_Calendar.MonthNames = var_Calendar.LocMonthNames;
var_Calendar.WeekDays = var_Calendar.LocWeekDays;
var_Calendar.AMPM = var_Calendar.LocAMPM;
```

X++ (Dynamics Ax 2009)

```
public void init()
{
    COM com_Calendar;
    anytype var_Calendar;
    ;
    super();
    var_Calendar = exschedule1.Calendar(); com_Calendar = var_Calendar;
    com_Calendar.FirstWeekDay(com_Calendar.LocFirstWeekDay());
    com_Calendar.MonthNames(com_Calendar.LocMonthNames());
    com_Calendar.WeekDays(com_Calendar.LocWeekDays());
    com_Calendar.AMPM(com_Calendar.LocAMPM());
}
```

Delphi 8 (.NET only)

```
with AxSchedule1 do
begin
with Calendar do
begin
FirstWeekDay := LocFirstWeekDay;
MonthNames := LocMonthNames;
WeekDays := LocWeekDays;
AMPM := LocAMPM;
end;
end
```

Delphi (standard)

```
with Schedule1 do
begin
with Calendar do
begin
FirstWeekDay := LocFirstWeekDay;
MonthNames := LocMonthNames;
WeekDays := LocWeekDays;
AMPM := LocAMPM;
end;
end
```

VFP

```
with thisform.Schedule1
with .Calendar
.FirstWeekDay = .LocFirstWeekDay
.MonthNames = .LocMonthNames
.WeekDays = .LocWeekDays
.AMPM = .LocAMPM
endwith
endwith
```

dBASE Plus

```
local oSchedule,var_Calendar
```

```
oSchedule = form.Activex1.nativeObject
var_Calendar = oSchedule.Calendar
var_Calendar.FirstWeekDay = var_Calendar.LocFirstWeekDay
var_Calendar.MonthNames = var_Calendar.LocMonthNames
var_Calendar.WeekDays = var_Calendar.LocWeekDays
var_Calendar.AMPM = var_Calendar.LocAMPM
```

XBasic (Alpha Five)

Dim oSchedule as P Dim var_Calendar as P

```
oSchedule = topparent:CONTROL_ACTIVEX1.activex
var_Calendar = oSchedule.Calendar
var_Calendar.FirstWeekDay = var_Calendar.LocFirstWeekDay
var_Calendar.MonthNames = var_Calendar.LocMonthNames
var_Calendar.WeekDays = var_Calendar.LocWeekDays
var_Calendar.AMPM = var_Calendar.LocAMPM
```

Visual Objects

local var_Calendar as ICalendar

var_Calendar := oDCOCX_Exontrol1:Calendar
var_Calendar:FirstWeekDay := var_Calendar:LocFirstWeekDay
var_Calendar:MonthNames := var_Calendar:LocMonthNames
var_Calendar:WeekDays := var_Calendar:LocWeekDays
var_Calendar:AMPM := var_Calendar:LocAMPM

PowerBuilder

OleObject oSchedule,var_Calendar oSchedule = ole_1.Object var_Calendar = oSchedule.Calendar var_Calendar.FirstWeekDay = var_Calendar.LocFirstWeekDay var_Calendar.MonthNames = var_Calendar.LocMonthNames var_Calendar.WeekDays = var_Calendar.LocWeekDays var_Calendar.MPM = var_Calendar.LocMPM

property Calendar.NonworkingDays as Long

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

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

By default, the NonworkingDays property is 65 (Saturday(s) and Sunday(s)). The <u>NonworkingDaysPattern</u> and the <u>NonworkingDaysColor</u> which defines the pattern and the color to show the non-working days. The <u>FirstWeekDay</u> property indicates the first day of the week. The <u>NonworkingTimes</u> collection defines the non-working time for days. The <u>NonworkingPatterns</u> collection holds the pattern to be shown when a non-working time is displayed. By default, the nonworking days are not highlighted in the schedule panel. In order, to highlight the non-working days in the schedule panel, you have to add at least one element to the NonworkingTimes collection as shown in the samples bellow. If the NonworkingDaysPattern property is exPatternEmpty or NonworkingDays property is 0 the non-working days are not highlighted in the calendar panel.

You can select the non-working week days in the following table (In Internet Explorer, you have to allow running the script on this page).

	Saturda	y Friday [·]	Thursday	Wednesday	Tuesday	Monday	Sunday
Value	64	32	16	8	4	2	1
Bit	\checkmark						\checkmark

Click the Bit row for non-working week days and the value for property is:

0

									Mon	Tue	Wed	Thu	Fri	Sat	Sun
4			lune	201	2			08:00 AM :30							
	Мо	Tu	We	Th	Fr	Sa	Su	09:00 AM	•••••						
22	28	29	30	31	1	2	3	10:00 AM							
23	4	5	6	7	8	9	10	:30	0.000.0000						
24	11	12	13	14	15	16	17	11:00 AM :30							
25	18	19	20	21	22	23	24	12:00 PM							
26	25	26	27	28	29	30	1	:30 01:00 PM							
27	2	3	4	5	6	7	8	:30							
								02:00 PM :30							
_			Тос	day)			03:00 PM :30							

The following samples sets the Sundays as being non-working and also shows it on the schedule panel as in the above screen shot:

```
With Schedule1
With .Calendar
.NonworkingDays = 1
.Selection = "value in (#6/14/2012#,#6/15/2012#,#6/16/2012#,#6/17/2012#)"
.FirstWeekDay = 1
End With
.NonworkingTimes.Add 1,"00:00","00:00",-1
End With
```

VB6

```
With Schedule1
With .Calendar
.NonworkingDays = 1
.Selection = "value in (#6/14/2012#,#6/15/2012#,#6/16/2012#,#6/17/2012#)"
.FirstWeekDay = exMonday
End With
.NonworkingTimes.Add 1,"00:00","00:00",-1
End With
```

VB.NET

```
With Exschedule1
With .Calendar
.NonworkingDays = 1
.Selection = "value in (#6/14/2012#,#6/15/2012#,#6/16/2012#,#6/17/2012#)"
.FirstWeekDay = exontrol.EXSCHEDULELib.WeekDayEnum.exMonday
End With
.NonworkingTimes.Add(1,"00:00","00:00",-1)
End With
```

VB.NET for /COM

```
With AxSchedule1
With .Calendar
.NonworkingDays = 1
.Selection = "value in (#6/14/2012#,#6/15/2012#,#6/16/2012#,#6/17/2012#)"
```

```
.FirstWeekDay = EXSCHEDULELib.WeekDayEnum.exMonday
End With
.NonworkingTimes.Add(1,"00:00","00:00",-1)
End With
```

C++

```
Copy and paste the following directives to your header file as
it defines the namespace 'EXSCHEDULELib' for the library: 'ExSchedule 1.0 Control
Library'
```

```
#import <ExSchedule.dll>
using namespace EXSCHEDULELib;
*/
EXSCHEDULELib::ISchedulePtr spSchedule1 = GetDlgItem(IDC_SCHEDULE1)-
>GetControlUnknown();
EXSCHEDULELib::ICalendarPtr var_Calendar = spSchedule1->GetCalendar();
var_Calendar->PutNonworkingDays(1);
var_Calendar->PutSelection("value in
(#6/14/2012#,#6/15/2012#,#6/16/2012#,#6/17/2012#)");
var_Calendar->PutFirstWeekDay(EXSCHEDULELib::exMonday);
spSchedule1->GetNonworkingTimes()->Add(L"1",L"00:00",L"00:00",-1);
```

C++ Builder

```
Exschedulelib_tlb::ICalendarPtr var_Calendar = Schedule1->Calendar;
var_Calendar->NonworkingDays = 1;
var_Calendar->set_Selection(TVariant("value in
(#6/14/2012#,#6/15/2012#,#6/16/2012#,#6/17/2012#)"));
var_Calendar->FirstWeekDay = Exschedulelib_tlb::WeekDayEnum::exMonday;
Schedule1->NonworkingTimes->Add(L"1",L"00:00",L"00:00",-1);
```

```
var_Calendar.NonworkingDays = 1;
var_Calendar.Selection = "value in
(#6/14/2012#,#6/15/2012#,#6/16/2012#,#6/17/2012#)";
var_Calendar.FirstWeekDay = exontrol.EXSCHEDULELib.WeekDayEnum.exMonday;
exschedule1.NonworkingTimes.Add(1.ToString(),"00:00","00:00",-1);
```

JavaScript

```
<OBJECT classid="clsid:9B09E13D-7A88-4299-9DBE-383380435377"
id="Schedule1"> </OBJECT>
<SCRIPT LANGUAGE="JScript">
var var_Calendar = Schedule1.Calendar;
var_Calendar.NonworkingDays = 1;
var_Calendar.Selection = "value in
(#6/14/2012#,#6/15/2012#,#6/16/2012#,#6/17/2012#)";
var_Calendar.FirstWeekDay = 1;
Schedule1.NonworkingTimes.Add(1,"00:00","00:00",-1);
</SCRIPT>
```

C# for /COM

```
EXSCHEDULELib.Calendar var_Calendar = axSchedule1.Calendar;
var_Calendar.NonworkingDays = 1;
var_Calendar.Selection = "value in
(#6/14/2012#,#6/15/2012#,#6/16/2012#,#6/17/2012#)";
var_Calendar.FirstWeekDay = EXSCHEDULELib.WeekDayEnum.exMonday;
axSchedule1.NonworkingTimes.Add(1.ToString(),"00:00","00:00",-1);
```

X++ (Dynamics Ax 2009)

```
public void init()
{
    COM com_Calendar;
    anytype var_Calendar;
```

```
var_Calendar = exschedule1.Calendar(); com_Calendar = var_Calendar;
    com_Calendar.NonworkingDays(1);
    com_Calendar.Selection("value in
(#6/14/2012#,#6/15/2012#,#6/16/2012#,#6/17/2012#)");
    com_Calendar.FirstWeekDay(1/*exMonday*/);
    exschedule1.NonworkingTimes().Add(1,"00:00","00:00",-1);
```

Delphi 8 (.NET only)

super();

```
with AxSchedule1 do
begin
with Calendar do
begin
NonworkingDays := 1;
Selection := 'value in (#6/14/2012#,#6/15/2012#,#6/16/2012#,#6/17/2012#)';
FirstWeekDay := EXSCHEDULELib.WeekDayEnum.exMonday;
end;
NonworkingTimes.Add(1,'00:00','00:00',-1);
end
```

Delphi (standard)

```
with Schedule1 do
begin
with Calendar do
begin
NonworkingDays := 1;
Selection := 'value in (#6/14/2012#,#6/15/2012#,#6/16/2012#,#6/17/2012#)';
FirstWeekDay := EXSCHEDULELib_TLB.exMonday;
end;
NonworkingTimes.Add(1,'00:00','00:00',-1);
end
```

```
with thisform.Schedule1
with .Calendar
.NonworkingDays = 1
.Selection = "value in (#6/14/2012#,#6/15/2012#,#6/16/2012#,#6/17/2012#)"
.FirstWeekDay = 1
endwith
.NonworkingTimes.Add(1,"00:00","00:00",-1)
endwith
```

dBASE Plus

local oSchedule,var_Calendar

```
oSchedule = form.Activex1.nativeObject
var_Calendar = oSchedule.Calendar
var_Calendar.NonworkingDays = 1
var_Calendar.Selection = "value in
(#6/14/2012#,#6/15/2012#,#6/16/2012#,#6/17/2012#)"
var_Calendar.FirstWeekDay = 1
oSchedule.NonworkingTimes.Add(Str(1),"00:00","00:00",-1)
```

XBasic (Alpha Five)

```
Dim oSchedule as P
Dim var_Calendar as P
oSchedule = topparent:CONTROL_ACTIVEX1.activex
var_Calendar = oSchedule.Calendar
var_Calendar.NonworkingDays = 1
var_Calendar.Selection = "value in
(#6/14/2012#,#6/15/2012#,#6/16/2012#,#6/17/2012#)"
var_Calendar.FirstWeekDay = 1
oSchedule.NonworkingTimes.Add(1,"00:00","00:00",-1)
```

Visual Objects

```
local var_Calendar as ICalendar
var_Calendar := oDCOCX_Exontrol1:Calendar
var_Calendar:NonworkingDays := 1
var_Calendar:Selection := "value in
(#6/14/2012#,#6/15/2012#,#6/16/2012#,#6/17/2012#)"
var_Calendar:FirstWeekDay := exMonday
oDCOCX_Exontrol1:NonworkingTimes:Add(AsString(1),"00:00","00:00",-1)
```

PowerBuilder

```
OleObject oSchedule,var_Calendar
```

```
oSchedule = ole_1.Object
var_Calendar = oSchedule.Calendar
var_Calendar.NonworkingDays = 1
var_Calendar.Selection = "value in
(#6/14/2012#,#6/15/2012#,#6/16/2012#,#6/17/2012#)"
var_Calendar.FirstWeekDay = 1
oSchedule.NonworkingTimes.Add(String(1),"00:00","00:00",-1)
```

property Calendar.NonworkingDaysColor as Color

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

Туре	Description
Color	A Color expression that indicates the color to show the non-working days in the calendar panel.

The NonworkingDaysColor property specifies the color being used to fill the non-working days. Use he <u>NonworkingDaysPattern</u> property to specify the brush to fill the nonworking days area. Use the <u>NonworkingDays</u> property to specify the nonworking days. Use the NonworkingDaysPattern property to specify the pattern to fill non-working days. If the NonworkingDaysPattern property is exPatternEmpty or NonworkingDays property is 0 the non-working days are not highlighted. If the <u>NonworkingDaysPattern</u> property includes the exPatternFrame, the <u>NonworkingDaysFrameColor</u> property indicates the color to show the frame around nonworking days.

property Calendar.NonworkingDaysFrameColor as Color

Retrieves or sets a value that indicates the color to show the non-working frame.

Туре	Description
Color	A Color expression that specifies the color to show the frame on non-working dates on the calendar panel.

If the <u>NonworkingDaysPattern</u> property includes the exPatternFrame, the NonworkingDaysFrameColor property indicates the color to show the frame around nonworking days. The NonworkingDaysFrameColor property has no effect if the <u>NonworkingDaysPattern</u> property includes NO exPatternFrame flag. Use the <u>NonworkingDays</u> property to specify the nonworking days. Use the NonworkingDaysPattern property to specify the pattern to fill non-working days. If the NonworkingDaysPattern property is exPatternEmpty or NonworkingDays property is 0 the non-working days are not highlighted. The <u>NonworkingDaysColor</u> property specifies the color being used to fill the non-working days.

property Calendar.NonworkingDaysPattern as PatternEnum

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

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

By default, the NonworkingDaysPattern property is exPatternBDiagonal. Use the NonworkingDaysPattern property to specify the brush to fill the nonworking days area. Use the NonworkingDays property to specify the nonworking days. Use the NonworkingDaysPattern property to specify the pattern to fill non-working days. If the NonworkingDaysPattern property is exPatternEmpty or NonworkingDays property is 0 the non-working days are not highlighted. The NonworkingDaysColor property specifies the color being used to fill the non-working days. If the NonworkingDaysPattern property includes the exPatternFrame, the NonworkingDaysFrameColor property indicates the color to show the frame around nonworking days.

property Calendar.OnSelectDate as OnSelectDateEnum

Specifies the action that the control does once the user selects new dates in the calendar panel.

Туре	Description
<u>OnSelectDateEnum</u>	An OnSelectDateEnum expression that specifies the operation to perform when user selects a date in the calendar panel.

By default, the OnSelectDate property is exFitSelToView, which indicates that the selected date in the calendar panel, is enlarged so it fit the schedule view. Use the OnSelectDate property to prevent changing the selected date in the schedule view, when user clicks or selects a new date in the calendar panel. The OnSelectDate property supports the following values:

- exFitSelToView (default), that indicates that the selected dates fits the schedule view. In this case the schedule view may be magnified or shrink and move to a new position. The <u>FitSelToView</u> property specifies the list of additional dates to be included in the schedule view, when selection is changed.
- exNoViewChange, indicates that no action is taken when user clicks or selects a date in the calendar panel
- exEnsureVisibleDate, ensures that the selected date fit the schedule view, without zooming the schedule view. This option does not zoom the schedule view. The <u>EnsureVisible</u> method ensures that giving date fits the schedule's view.

The <u>DayViewWidth</u> property specifies the width, in pixels, of the date in the schedule panel. The <u>DayViewOffsetX</u> property indicates the horizontal scroll position of the schedule's view. The <u>DayViewHeight</u> property specifies the height, in pixels, of the date in the schedule panel. The <u>DayViewOffsetY</u> property indicates the vertical scroll position of the schedule's view.

property Calendar.Parent as Long

Specifies the handle of the window that hosts the calendar panel.

Туре	Description
Long	A long expression that specifies the handle of the window that hosts the calendar panel.

By default, the Parent property is 0, which indicates that the calendar is hosted by the scheduler itself. Use the Parent property to move the calendar panel outside of the scheduler. The Parent property retrieves the handle of the window that hosts the calendar panel. If the Parent property is non-zero, the calendar panel fits the window's host client area. You can call or set the Parent property multiple time, and if necessary the calendar panel is resized to fit the new window space. In other words, if you resize the window that hosts the calendar panel is resized so it fits the host's client area. By default, the calendar panel of the component can be placed to the left or right of the component. For instance, the calendar panel can not be placed on the top or bottom side of the component, so in this case you can use the Parent property to place the calendar panel anywhere on your form/dialog.

If setting the Parent property to a

- zero value, the calendar is re-attached to the scheduler, so it becomes internal. In this case the <u>OnResizeControl</u> property specifies the position of the calendar panel, the auto-hide option, and so on.
- **non-zero handle**, it indicates the handle of the window that will display or host the calendar panel. In this case, the calendar panel becomes external. The OnResizeControl property has no effect, when the calendar panel is external. The value being passed to the Parent property usually comes from a hWnd property.

Here's how you can place the scheduler and the calendar to different places:

- Insert two eXSchedule components to the same form, with names: Schedule1 and Schedule2
- Handle the Load event of the form/dialog and call the following code:

```
Private Sub Form_Load()
Schedule1.Calendar.Parent = Schedule2.hWnd
End Sub
```

This way the second scheduler component acts as a host for the calendar panel of the first schedule component. Any action on the schedule or calendar will be reflected on both. When a window hosts the calendar panel, it fits the entire client area. In case you re-size

the window that hosts the calendar panel, you can re-assign the Calendar.Parent property, so the calendar panel updates its size so it fits the new client area of the host.

property Calendar.SelCount as Long

Indicates the number of dates being selected in the calendar panel.

Туре	Description
Long	A long expression that specifies the number of selected dates in the calendar panel.

The SelCount property counts the dates being selected in the calendar panel. The <u>SingleSel</u> property indicates whether the user can select one or multiple dates. If the SingleSel property is True, the SelCount property always returns 1. The <u>SelDate</u> property can be used to get the selected date giving its index in the selection dates collection. The <u>AllowSelectDate</u> property indicates the keys combination so the user can select new dates in the calendar panel, and so, new dates to be shown in the schedule view. Once the user starts selecting a new date in the calendar panel, the control fires the <u>LayoutStartChanging</u>(exCalendarSelectionChange). Once a new date is selected, the <u>LayoutEndChanging</u>(exCalendarSelectionChange) event occurs.

You can use the SelCount/<u>SelDate</u> or <u>Selection</u> to enumerate the selected dates. Use the <u>Selection/SelectDate</u> property to change programmatically the dates being selected in the calendar, including the dates to be shown in the schedule view.

The following VB sample shows how you can enumerate the selected dates using the SelCount and SelDate properties once the selection is changed:

```
Private Sub Schedule1_LayoutEndChanging(ByVal Operation As
EXSCHEDULELibCtl.LayoutChangingEnum)
If Operation = exCalendarSelectionChange Then
Dim i As Long
With Schedule1.Calendar
For i = 0 To .SelCount() - 1
Debug.Print "Select: " & .SelDate(i)
Next
End With
End If
End Sub
```

The following VB/NET sample shows how you can enumerate the selected dates using the SelCount and SelDate properties once the selection is changed:

Private Sub Exschedule1_LayoutEndChanging(ByVal sender As System.Object, ByVal Operation As exontrol.EXSCHEDULELib.LayoutChangingEnum) Handles

```
Exschedule1.LayoutEndChanging

If Operation =

exontrol.EXSCHEDULELib.LayoutChangingEnum.exCalendarSelectionChange Then

Dim i As Long = 0

With Exschedule1.Calendar

For i = 0 To .SelCount - 1

Debug.Print("Select: " & .get_SelDate(i))

Next

End With

End If

End Sub
```

The following C# sample shows how you can enumerate the selected dates using the SelCount and SelDate properties once the selection is changed:

```
private void exschedule1_LayoutEndChanging(object sender,
exontrol.EXSCHEDULELib.LayoutChangingEnum Operation)
{
    if ( Operation ==
    exontrol.EXSCHEDULELib.LayoutChangingEnum.exCalendarSelectionChange )
    {
      for (int i = 0; i < exschedule1.Calendar.SelCount; i++)
        System.Diagnostics.Debug.Print("Select: " +
      exschedule1.Calendar.get_SelDate(i).ToString());
    }
}
```

The following VFP sample shows how you can enumerate the selected dates using the SelCount and SelDate properties once the selection is changed:

```
*** ActiveX Control Event ***
LPARAMETERS operation
* 1 ' exCalendarSelectionChange
If Operation = 1 Then
for i = 0 to thisform.Schedule1.Calendar.SelCount() - 1
wait window TToC(thisform.Schedule1.Calendar.SelDate(i))
next
EndIf
```

The following C++ sample shows how you can enumerate the selected dates using the SelCount and SelDate properties once the selection is changed:

```
void LayoutEndChangingSchedule1(long Operation)
{
    if ( Operation == EXSCHEDULELib::exCalendarSelectionChange )
    {
        for ( int i = 0; i < m_spSchedule->Calendar->SelCount; i++ )
        {
            CString sMessage;
            sMessage.Format(_T("Select: %f\r\n"), m_spSchedule->Calendar->SelDate[i] );
            OutputDebugString( sMessage );
        }
    }
}
```

where m_spSchedule is of EXSCHEDULELib::ISchedulePtr type.

property Calendar.SelDate (Index as Long) as Date

Gets the date being selected giving its index in the selection.

Туре	Description
Index as Long	A Long expression that specifies the index of the selected date to be retrieved.
Date	A DATE expression that specifies the selected date, or ZERO, if index is not correct.

The SelDate property can be used to get the selected date giving its index in the selection dates collection. The <u>SelCount</u> property counts the dates being selected in the calendar panel. The <u>SingleSel</u> property indicates whether the user can select one or multiple dates. If the SingleSel property is True, the SelCount property always returns 1. IN this case, you can always use the SelDate(0) to get the selected date. The <u>AllowSelectDate</u> property indicates the keys combination so the user can select new dates in the calendar panel, and so, new dates to be shown in the schedule view. Once the user starts selecting a new date in the calendar panel, the control fires the

<u>LayoutStartChanging</u>(exCalendarSelectionChange). Once a new date is selected, the <u>LayoutEndChanging</u>(exCalendarSelectionChange) event occurs. The <u>Select</u> method can be used to select by code the current month, current week, current week day and the current/focus day.

You can use the <u>SelCount</u>/SelDate or <u>Selection</u> to enumerate the selected dates. Use the <u>Selection/SelectDate</u> property to change programmatically the dates being selected in the calendar, including the dates to be shown in the schedule view.

The following VB sample shows how you can enumerate the selected dates using the SelCount and SelDate properties once the selection is changed:

```
Private Sub Schedule1_LayoutEndChanging(ByVal Operation As
EXSCHEDULELibCt1.LayoutChangingEnum)
If Operation = exCalendarSelectionChange Then
Dim i As Long
With Schedule1.Calendar
For i = 0 To .SelCount() - 1
Debug.Print "Select: " & .SelDate(i)
Next
End With
End If
End Sub
```

The following VB/NET sample shows how you can enumerate the selected dates using the SelCount and SelDate properties once the selection is changed:

```
Private Sub Exschedule1_LayoutEndChanging(ByVal sender As System.Object, ByVal
Operation As exontrol.EXSCHEDULELib.LayoutChangingEnum) Handles
Exschedule1.LayoutEndChanging
If Operation =
exontrol.EXSCHEDULELib.LayoutChangingEnum.exCalendarSelectionChange Then
Dim i As Long = 0
With Exschedule1.Calendar
For i = 0 To .SelCount - 1
Debug.Print("Select: " & .get_SelDate(i))
Next
End With
End If
End Sub
```

The following C# sample shows how you can enumerate the selected dates using the SelCount and SelDate properties once the selection is changed:

```
private void exschedule1_LayoutEndChanging(object sender,
exontrol.EXSCHEDULELib.LayoutChangingEnum Operation)
{
    if ( Operation ==
    exontrol.EXSCHEDULELib.LayoutChangingEnum.exCalendarSelectionChange )
    {
      for (int i = 0; i < exschedule1.Calendar.SelCount; i++)
        System.Diagnostics.Debug.Print("Select: " +
      exschedule1.Calendar.get_SelDate(i).ToString());
    }
}
```

The following VFP sample shows how you can enumerate the selected dates using the SelCount and SelDate properties once the selection is changed:

```
*** ActiveX Control Event ***
LPARAMETERS operation
* 1 ' exCalendarSelectionChange
If Operation = 1 Then
```

```
for i = 0 to thisform.Schedule1.Calendar.SelCount() - 1
    wait window TToC(thisform.Schedule1.Calendar.SelDate(i))
    next
EndIf
```

The following C++ sample shows how you can enumerate the selected dates using the SelCount and SelDate properties once the selection is changed:

```
void LayoutEndChangingSchedule1(long Operation)
{
    if ( Operation == EXSCHEDULELib::exCalendarSelectionChange )
    {
        for ( int i = 0; i < m_spSchedule->Calendar->SelCount; i++ )
        {
            CString sMessage;
            sMessage.Format(_T("Select: %f\r\n"), m_spSchedule->Calendar->SelDate[i] );
            OutputDebugString( sMessage );
        }
    }
}
```

where m_spSchedule is of EXSCHEDULELib::ISchedulePtr type.

method Calendar.Select (newVal as SelectCalendarDateEnum)

Select the current (focus) day, week, month, year or week day in the calendar panel.

Туре	Description
newVal as <u>SelectCalendarDateEnum</u>	A SelectCalendarDate expression that specifies the selection should be made.

The Select method can be used to select by code/programmatically the current year, month, week, week day and the current/focus day. The FocusDate property indicates the current or the focused day. The Select method does not change the FocusDate property. The user can change the focused date using the mouse or the keyboard. Also, the Selection or the SelectDate changes programatically the selection/focused date by code. The LayoutEndChanging(exCalendarSelectionChange) event occurs once the Select method is invoked.

property Calendar.SelectDate(Date as Date) as Boolean

Selects or unselects a date in the calendar panel.

Туре	Description
Date as Date	A DATE expression to be selected or unselected
Boolean	A Boolean expression that specifies whether the date is selected (True) or unselected (False)

The SelectDate property can be used to programmatically select or unselect the giving date. An alternative to SelectDate is using the <u>Selection</u> that can uses an expression to indicate the dates to be selected. The <u>Selection</u> = "0" unselects all dates in the calendar panel. The <u>Select</u> method can be used to select by code the current month, current week, current week day and the current/focus day.

The <u>SelCount</u> property counts the dates being selected in the calendar panel. The <u>SingleSel</u> property indicates whether the user can select one or multiple dates. If the SingleSel property is True, the SelCount property always returns 1. The <u>SelDate</u> property can be used to get the selected date giving its index in the selection dates collection. The <u>AllowSelectDate</u> property indicates the keys combination so the user can select new dates in the calendar panel, and so, new dates to be shown in the schedule view. Once the user starts selecting a new date in the calendar panel, the control fires the <u>LayoutStartChanging</u>(exCalendarSelectionChange). Once a new date is selected, the <u>LayoutEndChanging</u>(exCalendarSelectionChange) event occurs. The DateFromPoint property indicates the date in the calendar panel from the cursor.

The following samples shows how programmatically you can select a single date:

VBA (MS Access, Excell...)

```
With Schedule1
With .Calendar
.Selection = "0"
.SelectDate(#1/1/2012#) = True
End With
End With
```

VB6

With Schedule1 With .Calendar .**Selection** = "0"

```
.SelectDate(#1/1/2012#) = True
```

End With End With

VB.NET

With Exschedule1 With .Calendar .Selection = "0" .set_SelectDate(#1/1/2012#,True) End With End With

VB.NET for /COM

With AxSchedule1 With .Calendar .Selection = "0" .SelectDate(#1/1/2012#) = True End With End With

C++

```
/*
   Copy and paste the following directives to your header file as
    it defines the namespace 'EXSCHEDULELib' for the library: 'ExSchedule 1.0 Control
Library'
   #import <ExSchedule.dll>
   using namespace EXSCHEDULELib;
*/
EXSCHEDULELib::ISchedulePtr spSchedule1 = GetDlgItem(IDC_SCHEDULE1)-
>GetControlUnknown();
EXSCHEDULELib::ICalendarPtr var_Calendar = spSchedule1->GetCalendar();
   var_Calendar->PutSelection("0");
   var_Calendar->PutSelectDate("1/1/2012",VARIANT_TRUE);
```

C++ Builder

Exschedulelib_tlb::ICalendarPtr var_Calendar = Schedule1->Calendar; var_Calendar->set_Selection(TVariant("0")); var_Calendar->set_SelectDate(TDateTime(2012,1,1).operator double(),true);

C#

exontrol.EXSCHEDULELib.Calendar var_Calendar = exschedule1.Calendar; var_Calendar.**Selection** = "0";

var_Calendar.**set_SelectDate**(Convert.ToDateTime("1/1/2012",System.Globalization.CuUS")),true);

JavaScript

```
<OBJECT classid="clsid:9B09E13D-7A88-4299-9DBE-383380435377"
id="Schedule1"></OBJECT>
<SCRIPT LANGUAGE="JScript">
var var_Calendar = Schedule1.Calendar;
var_Calendar.Selection = "0";
var_Calendar.SelectDate("1/1/2012") = true;
</SCRIPT>
```

C# for /COM

```
EXSCHEDULELib.Calendar var_Calendar = axSchedule1.Calendar;
var_Calendar.Selection = "0";
```

var_Calendar.set_SelectDate(Convert.ToDateTime("1/1/2012",System.Globalization.Cu
US")),true);

X++ (Dynamics Ax 2009)

public void init()

```
{
   COM com_Calendar;
   anytype var_Calendar;
   ;
   super();
   var_Calendar = exschedule1.Calendar(); com_Calendar = var_Calendar;
      com_Calendar.Selection("0");
      com_Calendar.SelectDate(str2Date("1/1/2012",213),true);
}
```

Delphi 8 (.NET only)

```
with AxSchedule1 do
begin
with Calendar do
begin
Selection := '0';
SelectDate['1/1/2012'] := True;
end;
end
```

Delphi (standard)

```
with Schedule1 do
begin
  with Calendar do
  begin
   Selection := '0';
   SelectDate['1/1/2012'] := True;
  end;
end
```

VFP

```
with thisform.Schedule1
with .Calendar
```

```
.Selection = "0"
.SelectDate({^2012-1-1}) = .T.
endwith
endwith
```

dBASE Plus

```
local oSchedule,var_Calendar
oSchedule = form.Activex1.nativeObject
var_Calendar = oSchedule.Calendar
var_Calendar.Selection = "0"
// var_Calendar.SelectDate("01/01/2012") = true
with (oSchedule)
TemplateDef = [Dim var_Calendar]
TemplateDef = var_Calendar
Template = [var_Calendar.SelectDate("01/01/2012") = true]
endwith
```

XBasic (Alpha Five)

```
Dim oSchedule as P
Dim var_Calendar as P
oSchedule = topparent:CONTROL_ACTIVEX1.activex
var_Calendar = oSchedule.Calendar
var_Calendar.Selection = "0"
'var_Calendar.SelectDate({01/01/2012}) = .t.
oSchedule.TemplateDef = "Dim var_Calendar"
oSchedule.TemplateDef = var_Calendar
oSchedule.TemplateDef = "var_Calendar.SelectDate(#01/01/2012#) = True"
```

Visual Objects

local var_Calendar as ICalendar

```
var_Calendar := oDCOCX_Exontrol1:Calendar
var_Calendar:Selection := "0"
var_Calendar:[SelectDate,SToD("20120101")] := true
```

PowerBuilder

OleObject oSchedule,var_Calendar

oSchedule = ole_1.Object
var_Calendar = oSchedule.Calendar
var_Calendar.Selection = "0"
var_Calendar.SelectDate(2012-01-01,true)

property Calendar.Selection as Variant

Returns or sets a safe array of selected dates in the calendar panel.

Туре	Description
Variant	A safe array of dates to be selected, or a string expression that indicates the date to be selected. When passing a string the Selection property supports value, operators and predefined functions as listed bellow

The Selection property of the Calendar object can be used to set or get the current selection (dates) in the control's calendar panel. The <u>Selection</u> property of control can be used to set or get the current selection (events) in the control's schedule panel.

The <u>SingleSel</u> property indicates whether the user can select one or multiple dates. As an alternative, you can use the <u>SelCount/SelDate</u> property to retrieve the collection of selected dates in the calendar panel. The <u>AllowSelectDate</u> property indicates the keys combination so the user can select new dates in the calendar panel, and so, new dates to be shown in the schedule view. Once the user starts selecting a new date in the calendar panel, the control fires the <u>LayoutStartChanging</u>(exCalendarSelectionChange). Once a new date is selected, the <u>LayoutEndChanging</u>(exCalendarSelectionChange) event occurs. The DateFromPoint property indicates the date in the calendar panel from the cursor. *The /NET and /WPF configurations provides the SelDates property equivalent of Selection, instead the SelDates returns a collection od DateTime objects (public virtual List<DateTime> SelDates).* The <u>ClipToSel</u> property indicates whether the control clips the schedule panel to view the selected dates only.

- The following <u>sample</u> shows how you can enumerate the selected dates, in the calendar panel, once the <u>LayoutEndChanging</u>(exCalendarSelectionChange) event occurs.
- The following <u>sample</u> shows how you can use an expression to select dates in the calendar panel.
- The Selection = "(int((yearday(value) -1- ((7-weekday(value yearday(value) + 1)) mod 7))/7) = int((yearday(date(``))-1)/7))" selects the current week.

The Selection property gets the selection as:

- a DATE expression, if the SingleSel property is True
- a safe or array of DATEs (collection), if the SingleSel property is True (VT_ARRAY | VT_VARIANT).

The Selection property sets the selection based on the SingleSel property as follows:

• if the SingleSel property is True, the Value being passes could be a VARIANT

expression that indicates the date to be selected. In other words, the value could be a DATE expression, as string expression that will be converted to a DATE, and so on. *Example: Selection = #1/1/2001#, which specifies that a single date is being selected*

- if the *SingleSel property is False*, the Value could be
 - a DATE expression, that indicates the newly selected date. Example: Selection = #1/1/2001#, which specifies that a single date is being selected
 - a safe array of DATE expressions, that indicates the newly selected dates. *Example: Selection = Array(#1/1/2001#,#1/2/2001#,#1/3/2001#), which* specifies that a collection of dates is being selected
 - a string expression that specifies the date to be selected. Example: Selection = " (int((yearday(value) -1- ((7-weekday(value - yearday(value) + 1)) mod 7))/7) = int((yearday(date(``))-1)/7))", which indicates an expression that determines the dates to be selected. In this particular sample, it selects the current week, so 7 days are being selected.

When using the string format (as Selection = "month(value) = 5"), the **value** keyword indicates the date being queried for selection, and the expression supports 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:

- $\circ~$ 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
- Itrim (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 following sample shows how you can enumerate the selected dates, in the calendar panel, once the LayoutEndChanging(exCalendarSelectionChange) event occurs.

VB

Private Sub Schedule1_LayoutEndChanging(ByVal Operation As EXSCHEDULELibCtl.LayoutChangingEnum) If Operation = exCalendarSelectionChange Then Dim d As Variant For Each d In Schedule1.Calendar.Selection Debug.Print "Select: " & d Next End If End Sub

VB/NET

Private Sub Exschedule1_LayoutEndChanging(ByVal sender As System.Object, ByVal Operation As exontrol.EXSCHEDULELib.LayoutChangingEnum) Handles Exschedule1.LayoutEndChanging If Operation = exontrol.EXSCHEDULELib.LayoutChangingEnum.exCalendarSelectionChange Then For Each d As DateTime In Exschedule1.Calendar.**SelDates** Debug.Print(d.ToString()) Next End If End Sub

or:

Private Sub Exschedule1_LayoutEndChanging(ByVal sender As System.Object, ByVal Operation As exontrol.EXSCHEDULELib.LayoutChangingEnum) Handles Exschedule1.LayoutEndChanging If Operation =

```
exontrol.EXSCHEDULELib.LayoutChangingEnum.exCalendarSelectionChange Then
For Each d As DateTime In Exschedule1.Calendar.Selection
Debug.Print(d.ToString())
Next
End If
```

C#

End Sub

```
private void exschedule1_LayoutEndChanging(object sender,
exontrol.EXSCHEDULELib.LayoutChangingEnum Operation)
{
    if ( Operation ==
exontrol.EXSCHEDULELib.LayoutChangingEnum.exCalendarSelectionChange )
    {
      foreach (DateTime d in exschedule1.Calendar.SelDates)
        System.Diagnostics.Debug.Print(d.ToString());
    }
```

or:

```
private void exschedule1_LayoutEndChanging(object sender,
exontrol.EXSCHEDULELib.LayoutChangingEnum Operation)
{
    if ( Operation ==
exontrol.EXSCHEDULELib.LayoutChangingEnum.exCalendarSelectionChange )
    {
      foreach (DateTime d in exschedule1.Calendar.Selection as Array)
        System.Diagnostics.Debug.Print(d.ToString());
    }
}
```

VFP

```
*** ActiveX Control Event ***
LPARAMETERS operation
* 1 ' exCalendarSelectionChange
```

```
If Operation = 1 Then
  for i = 0 to thisform.Schedule1.Calendar.SelCount() - 1
    wait window TToC(thisform.Schedule1.Calendar.SelDate(i))
    next
EndIf
```

C++

```
void LayoutEndChangingSchedule1(long Operation)
{
  if (Operation = = EXSCHEDULELib::exCalendarSelectionChange)
  {
    _variant_t selection = m_spSchedule->Calendar->Selection;
    if (V_VT( &selection ) = = (VT_ARRAY | VT_VARIANT ) )
    {
      BYTE* p = NULL;
      long nCount = 0;
      if (SUCCEEDED(SafeArrayGetUBound(V_ARRAY(&selection), 1, &nCount)))
      {
        if (SUCCEEDED(SafeArrayAccessData(V_ARRAY(&selection), (LPVOID*)&p)))
        {
          for (long i = 0; i < nCount + 1; i++, p += sizeof(VARIANT))
           {
             VARIANT* pValue = (VARIANT*)p;
             if (V_VT(pValue) == VT_DATE)
             {
               CString strMessage;
               strMessage.Format( _T("Select: %f\r\n"), V_DATE( pValue ) );
               OutputDebugString( strMessage );
             }
           SafeArrayUnaccessData( V_ARRAY( & selection ) );
        }
      }
    }
```

where m_spSchedule is of EXSCHEDULELib::ISchedulePtr type.

The following sample selects the last three days (today, yesterday, and a day before):

VBA (MS Access, Excell...)

```
With Schedule1

With .Calendar

.NonworkingDays = 0

.ShowNonMonthDays = False

.FirstWeekDay = 0

.Selection = "(int(date(``)) - value) in (1,2,0)"

End With

.BorderSelStyle = -1

.Background(81) = RGB(240,240,240)

End With
```

VB6

With Schedule1	
With .Calendar	
.NonworkingDays = 0	
.ShowNonMonthDays = False	
.FirstWeekDay = exSunday	
. Selection = "(int(date(``)) - value) in (1,2,0)"	
End With	
.BorderSelStyle = exNoLines	
.Background(exScheduleMarkTodayBackColor) = RGB(240,240,240)	
End With	

VB.NET

```
With Exschedule1

With .Calendar

.NonworkingDays = 0

.ShowNonMonthDays = False

.FirstWeekDay = exontrol.EXSCHEDULELib.WeekDayEnum.exSunday

.Selection = "(int(date(``)) - value) in (1,2,0)"

End With
```

```
.BorderSelStyle = exontrol.EXSCHEDULELib.LinesStyleEnum.exNoLines
```

.set_Background(exontrol.EXSCHEDULELib.BackgroundPartEnum.exScheduleMarkToday

End With

VB.NET for /COM

```
With AxSchedule1

With .Calendar

.NonworkingDays = 0

.ShowNonMonthDays = False

.FirstWeekDay = EXSCHEDULELib.WeekDayEnum.exSunday

.Selection = "(int(date(``)) - value) in (1,2,0)"

End With

.BorderSelStyle = EXSCHEDULELib.LinesStyleEnum.exNoLines
```

.set_Background(EXSCHEDULELib.BackgroundPartEnum.exScheduleMarkTodayBackCol

End With

C++

```
/*

Copy and paste the following directives to your header file as

it defines the namespace 'EXSCHEDULELib' for the library: 'ExSchedule 1.0 Control

Library'

#import <ExSchedule.dll>

using namespace EXSCHEDULELib;

*/

EXSCHEDULELib::ISchedulePtr spSchedule1 = GetDlgItem(IDC_SCHEDULE1)-

>GetControlUnknown();

EXSCHEDULELib::ICalendarPtr var_Calendar = spSchedule1->GetCalendar();

var_Calendar->PutNonworkingDays(0);

var_Calendar->PutShowNonMonthDays(VARIANT_FALSE);

var_Calendar->PutFirstWeekDay(EXSCHEDULELib::exSunday);

var_Calendar->PutSleection("(int(date(``)) - value) in (1,2,0)");
```

```
spSchedule1->PutBorderSelStyle(EXSCHEDULELib::exNoLines);
spSchedule1-
```

> PutBackground(EXSCHEDULELib::exScheduleMarkTodayBackColor,RGB(240,240,240));

C++ Builder

```
Exschedulelib_tlb::ICalendarPtr var_Calendar = Schedule1->Calendar;
var_Calendar->NonworkingDays = 0;
var_Calendar->ShowNonMonthDays = false;
var_Calendar->FirstWeekDay = Exschedulelib_tlb::WeekDayEnum::exSunday;
var_Calendar->set_Selection(TVariant("(int(date(``)) - value) in (1,2,0)"));
Schedule1->BorderSelStyle = Exschedulelib_tlb::LinesStyleEnum::exNoLines;
Schedule1-
>Background[Exschedulelib_tlb::BackgroundPartEnum::exScheduleMarkTodayBackColo
= RGB(240,240,240);
```

C#

```
exontrol.EXSCHEDULELib.Calendar var_Calendar = exschedule1.Calendar;
var_Calendar.NonworkingDays = 0;
var_Calendar.ShowNonMonthDays = false;
var_Calendar.FirstWeekDay = exontrol.EXSCHEDULELib.WeekDayEnum.exSunday;
var_Calendar.Selection = "(int(date(`)) - value) in (1,2,0)";
exschedule1.BorderSelStyle = exontrol.EXSCHEDULELib.LinesStyleEnum.exNoLines;
exschedule1.set_Background(exontrol.EXSCHEDULELib.BackgroundPartEnum.exSchedule1.
```

JavaScript

```
<OBJECT classid="clsid:9B09E13D-7A88-4299-9DBE-383380435377"
id="Schedule1"></OBJECT>
```

```
<SCRIPT LANGUAGE="JScript">
var var_Calendar = Schedule1.Calendar;
```

```
var_Calendar.NonworkingDays = 0;
var_Calendar.ShowNonMonthDays = false;
var_Calendar.FirstWeekDay = 0;
var_Calendar.Selection = "(int(date(``)) - value) in (1,2,0)";
Schedule1.BorderSelStyle = -1;
Schedule1.Background(81) = 15790320;
</SCRIPT>
```

C# for /COM

```
EXSCHEDULELib.Calendar var_Calendar = axSchedule1.Calendar;
var_Calendar.NonworkingDays = 0;
var_Calendar.ShowNonMonthDays = false;
var_Calendar.FirstWeekDay = EXSCHEDULELib.WeekDayEnum.exSunday;
var_Calendar.Selection = "(int(date(``)) - value) in (1,2,0)";
axSchedule1.BorderSelStyle = EXSCHEDULELib.LinesStyleEnum.exNoLines;
axSchedule1.set_Background(EXSCHEDULELib.BackgroundPartEnum.exScheduleMarkTc
(uint)ColorTranslator.ToWin32(Color.FromArgb(240,240,240)));
```

X++ (Dynamics Ax 2009)

```
public void init()
{
    COM com_Calendar;
    anytype var_Calendar;
    anytype var_Calendar;
    ;
    super();
    var_Calendar = exschedule1.Calendar(); com_Calendar = var_Calendar;
    com_Calendar.NonworkingDays(0);
    com_Calendar.ShowNonMonthDays(false);
    com_Calendar.FirstWeekDay(0/*exSunday*/);
    com_Calendar.Selection("(int(date(``)) - value) in (1,2,0)");
    exschedule1.BorderSelStyle(-1/*exNoLines*/);
```

exschedule1.Background(81/*exScheduleMarkTodayBackColor*/,WinApi::RGB2int(240,2

Delphi 8 (.NET only)

```
with AxSchedule1 do
begin
with Calendar do
begin
NonworkingDays := 0;
ShowNonMonthDays := False;
FirstWeekDay := EXSCHEDULELib.WeekDayEnum.exSunday;
Selection := '(int(date(``)) - value) in (1,2,0)';
end;
BorderSelStyle := EXSCHEDULELib.LinesStyleEnum.exNoLines;
set_Background(EXSCHEDULELib.BackgroundPartEnum.exScheduleMarkTodayBackColc
```

Delphi (standard)

end

```
with Schedule1 do
begin
with Calendar do
begin
NonworkingDays := 0;
ShowNonMonthDays := False;
FirstWeekDay := EXSCHEDULELib_TLB.exSunday;
Selection := '(int(date(``)) - value) in (1,2,0)';
end;
BorderSelStyle := EXSCHEDULELib_TLB.exNoLines;
Background[EXSCHEDULELib_TLB.exScheduleMarkTodayBackColor] := $f0f0f0;
end
```

VFP

with thisform.Schedule1

}

```
with .Calendar
.NonworkingDays = 0
.ShowNonMonthDays = .F.
.FirstWeekDay = 0
.Selection = "(int(date(``)) - value) in (1,2,0)"
endwith
.BorderSelStyle = -1
.Object.Background(81) = RGB(240,240,240)
endwith
```

dBASE Plus

local oSchedule,var_Calendar

```
oSchedule = form.Activex1.nativeObject
var_Calendar = oSchedule.Calendar
var_Calendar.NonworkingDays = 0
var_Calendar.ShowNonMonthDays = false
var_Calendar.FirstWeekDay = 0
var_Calendar.Selection = "(int(date(``)) - value) in (1,2,0)"
oSchedule.BorderSelStyle = -1
oSchedule.Template = [Background(81) = 0xf0f0f0] // oSchedule.Background(81)
= 0xf0f0f0
```

XBasic (Alpha Five)

```
Dim oSchedule as P
Dim var_Calendar as P
oSchedule = topparent:CONTROL_ACTIVEX1.activex
var_Calendar = oSchedule.Calendar
var_Calendar.NonworkingDays = 0
var_Calendar.ShowNonMonthDays = .f.
var_Calendar.FirstWeekDay = 0
var_Calendar.FirstWeekDay = 0
var_Calendar.Selection = "(int(date(``)) - value) in (1,2,0)"
oSchedule.BorderSelStyle = -1
oSchedule.Template = "Background(81) = 15790320" ' oSchedule.Background(81)
```

```
= 15790320
```

Visual Objects

local var_Calendar as ICalendar

```
var_Calendar := oDCOCX_Exontrol1:Calendar
var_Calendar:NonworkingDays := 0
var_Calendar:ShowNonMonthDays := false
var_Calendar:FirstWeekDay := exSunday
var_Calendar:Selection := "(int(date(``)) - value) in (1,2,0)"
oDCOCX_Exontrol1:BorderSelStyle := exNoLines
oDCOCX_Exontrol1:[Background,exScheduleMarkTodayBackColor] :=
RGB(240,240,240)
```

PowerBuilder

```
OleObject oSchedule,var_Calendar
```

```
oSchedule = ole_1.Object
var_Calendar = oSchedule.Calendar
var_Calendar.NonworkingDays = 0
var_Calendar.ShowNonMonthDays = false
var_Calendar.FirstWeekDay = 0
var_Calendar.Selection = "(int(date(``)) - value) in (1,2,0)"
oSchedule.BorderSelStyle = -1
oSchedule.Background(81,RGB(240,240,240))
```

property Calendar.ShortDateFormat as String

Indicates the short date format.

Туре	Description
String	A String expression that defines the short date format. The ShortDateFormat suports tags as described bellow.

By default, the ShortDateFormat property is "<%loc_sdate%>", so the format of the long date as defined in the regional settings is currently used. The

KnownProperty(exEventDisplayShortMargins) property uses the ShortDateFormat property to display the event's margins in a short date format. For instance, an all-day event (<u>AllDayEvent</u> property) can display the starting and ending margins of the event in a short date format. The LongDateFormat property defines the long date format to be used when label properties includes the <%=%257%> TAG.

In conclusion, the ShortDateFormat property defines the long date format being used to display the event's margins, when the <%**=%256**%> is included in the *label* properties such as:

- <u>DefaultEventLongLabel</u>, defines the HTML labels for events, when it fit entirely in the event's body.
- <u>DefaultEventShortLabel</u>, defines the labels for events (no HTML attribute is applied), when it does not fit the event's body
- CreateEventLabel, defines the label when creating a new event by dragging
- UpdateEventsLabel, defines the label of the events being moved or resized at runtime
- Event.<u>ShortLabel</u>, defines the event's short label, or the label to be shown when the LongLabel does not fit entirely the event's body.
- Event.<u>LongLabel</u>, defines the event's HTML long label, when it fits the body. If the LongLabel does not fit entirely the event's body, the ShortLabel is displayed instead.
- Event.<u>ExtraLabel</u>, defines the event's extra HTML label. The event's ExtraLabel is displayed ONLY, if the LongLabel fits the event's body

The property supports the following TAGs:

- <%d%> Day of the month in one or two numeric digits, as needed (1 to 31).
- <%dd%> Day of the month in two numeric digits (01 to 31).
- <%d1%> First letter of the weekday (S to S). (Use the <u>WeekDays</u> property to specify the name of the days in the week)
- <%d2%> First two letters of the weekday (Su to Sa). (Use the <u>WeekDays</u> property to specify the name of the days in the week)
- <%d3%> First three letters of the weekday (Sun to Sat). (Use the <u>WeekDays</u> property to specify the name of the days in the week)

- <%ddd%> First three letters of the weekday (Sun to Sat). (Use the <u>WeekDays</u> property to specify the name of the days in the week). You can use the <%loc_ddd%> that indicates the day of week as a three-letter abbreviation using the current user regional and language settings.
- <%loc_ddd%> Indicates the day of week as a three-letter abbreviation using the current user regional and language settings.
- <%dddd%> Full name of the weekday (Sunday to Saturday). (Use the <u>WeekDays</u> property to specify the name of the days in the week). You can use the <%loc_dddd%> that indicates day of week as its full name using the current user regional and language settings.
- <%loc_dddd%> Indicates day of week as its full name using the current user regional and language settings.
- <%i%> Displays the number instead the date. For instance, you can display numbers as 1000, 1001, 1002, 1003, instead dates. (the valid range is from -647,434 to 2,958,465)
- <%**w**%> Day of the week (1 to 7).
- <%**ww**%> Week of the year (1 to 53).
- <%m%> Month of the year in one or two numeric digits, as needed (1 to 12).
- <%mr%> Month of the year in Roman numerals, as needed (I to XII).
- <%mm%> Month of the year in two numeric digits (01 to 12).
- <%m1%> First letter of the month (J to D). (Use the <u>MonthNames</u> property to specify the name of the months in the year)
- <%m2%> First two letters of the month (Ja to De). (Use the <u>MonthNames</u> property to specify the name of the months in the year)
- <%m3%> First three letters of the month (Jan to Dec). (Use the <u>MonthNames</u> property to specify the name of the months in the year)
- <%mmm%> First three letters of the month (Jan to Dec). (Use the MonthNames property to specify the name of the months in the year). You can use the <%loc_mmm%> that indicates month as a three-letter abbreviation using the current user regional and language settings.
- <%loc_mmm%> Indicates month as a three-letter abbreviation using the current user regional and language settings.
- <%mmm%> Full name of the month (January to December). (Use the MonthNames property to specify the name of the months in the year). You can use the <%loc_mmm%> that indicates month as its full name using the current user regional and language settings.
- <%loc_mmmm%> Indicates month as its full name using the current user regional and language settings.
- <%q%> Date displayed as the quarter of the year (1 to 4).
- $\langle \mathbf{y} \mathbf{y} \rangle$ Number of the day of the year (1 to 366).
- <%yy%> Last two digits of the year (01 to 99).
- <%**yyyy**%> Full year (0100 to 9999).

- <%hy%> Date displayed as the half of the year (1 to 2).
- <%loc_gg%> Indicates period/era using the current user regional and language settings.
- <%loc_sdate%> Indicates the date in the short format using the current user regional and language settings.
- <%loc_ldate%> Indicates the date in the long format using the current user regional and language settings.
- <%loc_dsep%> Indicates the date separator using the current user regional and language settings (/).
- <%h%> Hour in one or two digits, as needed (0 to 23).
- <%**hh**%> Hour in two digits (00 to 23).
- <%n%> Minute in one or two digits, as needed (0 to 59).
- <%**nn**%> Minute in two digits (00 to 59).
- <%**s**%> Second in one or two digits, as needed (0 to 59).
- <%**ss**%> Second in two digits (00 to 59).
- <%AM/PM%> Twelve-hour clock with the uppercase letters "AM" or "PM", as appropriate. (Use the <u>AMPM</u> property to specify the name of the AM and PM indicators). You can use the <%loc_AM/PM%> that indicates the time marker such as AM or PM using the current user regional and language settings. You can use <%loc_A/P%> that indicates the one character time marker such as A or P using the current user regional and language settings.
- <%loc_AM/PM%> Indicates the time marker such as AM or PM using the current user regional and language settings.
- <%loc_A/P%> Indicates the one character time marker such as A or P using the current user regional and language settings.
- <%loc_time%> Indicates the time using the current user regional and language settings.
- <%loc_time24%> Indicates the time in 24 hours format without a time marker using the current user regional and language settings.
- <%loc_tsep%> indicates the time separator using the current user regional and language settings (:).

The following tags are displayed based on the user's Regional and Language Options:

- <%loc_sdate%> Indicates the date in the short format using the current user settings.
- <%loc_ldate%> Indicates the date in the long format using the current user settings.
- <%loc_ddd%> Indicates day of week as a three-letter abbreviation using the current user settings.
- <%loc_dddd%> Indicates day of week as its full name using the current user settings.
- <%loc_mmm%> Indicates month as a three-letter abbreviation using the current user

settings.

- <%loc_mmmm%> Indicates month as its full name using the current user settings.
- <%loc_gg%> Indicates period/era using the current user settings.
- <%loc_dsep%> Indicates the date separator using the current user settings.
- <%loc_time%> Indicates the time using the current user settings.
- <%loc_time24%> Indicates the time in 24 hours format without a time marker using the current user settings.
- <%loc_AM/PM%> Indicates the time marker such as AM or PM using the current user settings.
- <%loc_A/P%> Indicates the one character time marker such as A or P using the current user settings.
- <%loc_tsep%> Indicates the time separator using the current user settings.
- <%loc_y%> Represents the Year only by the last digit, using current regional settings.
- <%loc_yy%> Represents the Year only by the last two digits, using current regional settings. A leading zero is added for single-digit years.
- <%loc_yyyy%> Represents the Year by a full four or five digits, depending on the calendar used. Thai Buddhist and Korean calendars have five-digit years. The "yyyy" pattern shows five digits for these two calendars, and four digits for all other supported calendars. Calendars that have single-digit or two-digit years, such as for the Japanese Emperor era, are represented differently. A single-digit year is represented with a leading zero, for example, "03". A two-digit year is represented with two digits, for example, "13". No additional leading zeros are displayed.

property Calendar.ShortTimeFormat as String

Indicates the short time format.

Туре	Description
String	A String expression that defines the short time format. The ShortTimeFormat supports tags as described bellow.

By default, the ShortTimeFormat property is "<%h%>:<%nn%> <%AM/PM%>", as 8:15 AM. The <u>KnownProperty</u>(exEventDisplayShortMargins) property uses the ShortTimeFormat property to display the event's margins in a short time format. The ShortTimeFormat property is used by label properties if <%=%256%> TAG is included as explained bellow. The <u>LongTimeFormat</u> property defines the long time format to be used when label properties includes the <%=%257%> TAG. The <u>MajorTimeLabel</u> property defines the format of the time to be displayed on the control's time scale.

The following screen shot shows the short time format using the AM/PM time indicators (by default, "<%h%>:<%nn%> <%AM/PM%>"):



The following screen shot shows the short time format using no AM/PM time indicators (" <%h%>:<%nn%>"):



In conclusion, the ShortTimeFormat property defines the short time format being used to display the event's margins, when the <%=%256%> is included in the *label* properties such as:

- <u>DefaultEventLongLabel</u>, defines the HTML labels for events, when it fit entirely in the event's body.
- <u>DefaultEventShortLabel</u>, defines the labels for events (no HTML attribute is applied), when it does not fit the event's body
- CreateEventLabel, defines the label when creating a new event by dragging
- <u>UpdateEventsLabel</u>, defines the label of the events being moved or resized at runtime
- Event.<u>ShortLabel</u>, defines the event's short label, or the label to be shown when the LongLabel does not fit entirely the event's body. The ShortLabel displays no HTML tags
- Event.<u>LongLabel</u>, defines the event's HTML long label, when it fits the body. If the LongLabel does not fit entirely the event's body, the ShortLabel is displayed instead.
- Event.<u>ExtraLabel</u>, defines the event's extra HTML label. The event's ExtraLabel is displayed ONLY, if the LongLabel fits the event's body

The property supports the following TAGs:

- <%d%> Day of the month in one or two numeric digits, as needed (1 to 31).
- <%dd%> Day of the month in two numeric digits (01 to 31).
- <%d1%> First letter of the weekday (S to S). (Use the <u>WeekDays</u> property to specify the name of the days in the week)
- <%d2%> First two letters of the weekday (Su to Sa). (Use the <u>WeekDays</u> property to specify the name of the days in the week)

- <%d3%> First three letters of the weekday (Sun to Sat). (Use the <u>WeekDays</u> property to specify the name of the days in the week)
- <%ddd%> First three letters of the weekday (Sun to Sat). (Use the <u>WeekDays</u> property to specify the name of the days in the week). You can use the <%loc_ddd%> that indicates the day of week as a three-letter abbreviation using the current user regional and language settings.
- <%loc_ddd%> Indicates the day of week as a three-letter abbreviation using the current user regional and language settings.
- <%dddd%> Full name of the weekday (Sunday to Saturday). (Use the <u>WeekDays</u> property to specify the name of the days in the week). You can use the <%loc_dddd%> that indicates day of week as its full name using the current user regional and language settings.
- <%loc_dddd%> Indicates day of week as its full name using the current user regional and language settings.
- <%i%> Displays the number instead the date. For instance, you can display numbers as 1000, 1001, 1002, 1003, instead dates. (the valid range is from -647,434 to 2,958,465)
- <%**w**%> Day of the week (1 to 7).
- <%ww%> Week of the year (1 to 53).
- <%m%> Month of the year in one or two numeric digits, as needed (1 to 12).
- <%mr%> Month of the year in Roman numerals, as needed (I to XII).
- <%mm%> Month of the year in two numeric digits (01 to 12).
- <%m1%> First letter of the month (J to D). (Use the <u>MonthNames</u> property to specify the name of the months in the year)
- <%m2%> First two letters of the month (Ja to De). (Use the <u>MonthNames</u> property to specify the name of the months in the year)
- <%m3%> First three letters of the month (Jan to Dec). (Use the <u>MonthNames</u> property to specify the name of the months in the year)
- <%mmm%> First three letters of the month (Jan to Dec). (Use the MonthNames property to specify the name of the months in the year). You can use the <%loc_mmm%> that indicates month as a three-letter abbreviation using the current user regional and language settings.
- <%loc_mmm%> Indicates month as a three-letter abbreviation using the current user regional and language settings.
- <%mmm%> Full name of the month (January to December). (Use the MonthNames property to specify the name of the months in the year). You can use the <%loc_mmm%> that indicates month as its full name using the current user regional and language settings.
- <%loc_mmmm%> Indicates month as its full name using the current user regional and language settings.
- <%**q**%> Date displayed as the quarter of the year (1 to 4).
- $\langle \mathbf{y} \mathbf{y} \rangle$ Number of the day of the year (1 to 366).

- <%**yy**%> Last two digits of the year (01 to 99).
- <%yyyy%> Full year (0100 to 9999).
- <%hy%> Date displayed as the half of the year (1 to 2).
- <%loc_gg%> Indicates period/era using the current user regional and language settings.
- <%loc_sdate%> Indicates the date in the short format using the current user regional and language settings.
- <%loc_ldate%> Indicates the date in the long format using the current user regional and language settings.
- <%loc_dsep%> Indicates the date separator using the current user regional and language settings (/).
- <%h%> Hour in one or two digits, as needed (0 to 23).
- <%**hh**%> Hour in two digits (00 to 23).
- <%**n**%> Minute in one or two digits, as needed (0 to 59).
- <%**nn**%> Minute in two digits (00 to 59).
- <%s%> Second in one or two digits, as needed (0 to 59).
- <%**ss**%> Second in two digits (00 to 59).
- <%AM/PM%> Twelve-hour clock with the uppercase letters "AM" or "PM", as appropriate. (Use the <u>AMPM</u> property to specify the name of the AM and PM indicators). You can use the <%loc_AM/PM%> that indicates the time marker such as AM or PM using the current user regional and language settings. You can use <%loc_A/P%> that indicates the one character time marker such as A or P using the current user regional and language settings.
- <%loc_AM/PM%> Indicates the time marker such as AM or PM using the current user regional and language settings.
- <%loc_A/P%> Indicates the one character time marker such as A or P using the current user regional and language settings.
- <%loc_time%> Indicates the time using the current user regional and language settings.
- <%loc_time24%> Indicates the time in 24 hours format without a time marker using the current user regional and language settings.
- <%loc_tsep%> indicates the time separator using the current user regional and language settings (:).

The following tags are displayed based on the user's Regional and Language Options:

- <%loc_sdate%> Indicates the date in the short format using the current user settings.
- <%loc_ldate%> Indicates the date in the long format using the current user settings.
- <%loc_ddd%> Indicates day of week as a three-letter abbreviation using the current user settings.
- <%loc_dddd%> Indicates day of week as its full name using the current user

settings.

- <%loc_mmm%> Indicates month as a three-letter abbreviation using the current user settings.
- <%loc_mmmm%> Indicates month as its full name using the current user settings.
- <%loc_gg%> Indicates period/era using the current user settings.
- <%loc_dsep%> Indicates the date separator using the current user settings.
- <%loc_time%> Indicates the time using the current user settings.
- <%loc_time24%> Indicates the time in 24 hours format without a time marker using the current user settings.
- <%loc_AM/PM%> Indicates the time marker such as AM or PM using the current user settings.
- <%loc_A/P%> Indicates the one character time marker such as A or P using the current user settings.
- <%loc_tsep%> Indicates the time separator using the current user settings.
- <%loc_y%> Represents the Year only by the last digit, using current regional settings.
- <%loc_yy%> Represents the Year only by the last two digits, using current regional settings. A leading zero is added for single-digit years.
- <%loc_yyyy%> Represents the Year by a full four or five digits, depending on the calendar used. Thai Buddhist and Korean calendars have five-digit years. The "yyyy" pattern shows five digits for these two calendars, and four digits for all other supported calendars. Calendars that have single-digit or two-digit years, such as for the Japanese Emperor era, are represented differently. A single-digit year is represented with a leading zero, for example, "03". A two-digit year is represented with two digits, for example, "13". No additional leading zeros are displayed.

property Calendar.ShowGridLines as LinesStyleEnum

Shows or hides the grid lines in the calendar panel.

Туре	Description
LinesStyleEnum	A LinesStyleEnum expression that specifies the grid lines to be shown in the calendar panel.

By default, the ShowGridLines property is exLinesDot. Use the ShowGridLines property to hide the calendar's grid lines. The <u>Background</u>(exCalendarGridLineColor) property specifies the color to show the calendar's grid lines.

property Calendar.ShowHighlightEvent as Boolean

Returns or sets a value that indicates whether the calendar panel highlights days that contain events.

Туре	Description
Boolean	A Boolean expression that specifies whether the calendar panel highlights the dates with events or appointments

By default, the ShowHighlightEvent property is true, which indicates that the dates with events or appointments appear as bold in the calendar panel. You can use the ShowHighlightEvent property to prevent highlighting the the dates with events or appointments. You can use the <u>HighlightEvent</u> object to highlight the dates with events or appointments in the calendar panel. The <u>GroupHighlightEvent</u> property specifies if events are highlighted using the HighlightEvent property (False), or using the <u>CalendarHighlightEvent</u> property of the <u>Group</u> that event belongs to (True). The <u>ScheduleHighlightEvent</u> property specifies the visual appearance of dates with events in the schedule panel.

Using the Highlight object a date with events can combine one or more of the following options:

- **bold**, **Bold** property renders as bold text
- *italic*, *Italic* property renders as italic text
- <u>underline</u>, <u>Underline</u> property underlines the text
- strikeout, StrikeOut property shows the text with a horizontal line through its center
- change the font SiZe, <u>FontSize</u> property indicates the size of the font to display the text
- change the **font**, using the <u>Font</u> property
- change the text's <u>foreground</u> color, using the <u>ForeColor</u> property
- change the text's background color, using the <u>BackColor</u> property
- shows a pattern using the <u>Pattern</u> property

property Calendar.ShowNonMonthDays as Boolean

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

Туре	Description
Boolean	A boolean expression that specifies whether dates that belongs to another months are displayed on the calendar panel.

By default, the ShowNonMonthDays property is True. You can use the ShowNonMonthDays property to hide dates that are not belonging to the browsing month.

The following screen shot shows May 2012, while 29, 30 are from April, and 1 - 9 from June, non month days, (ShowNonMonthDays is True, by default):

	Su	Мо	Tu	We	Th	Fr	Sa
18	29	30	1	2	3	4	5
19	6	7	8	9	10	11	12
20	13	14	15	16	17	18	19
21	20	21	22	23	24	25	26
22	27	28	29	30	31	1	2
23	3	4	5	6	7	8	9

The following screen shot shows only May 2012, (ShowNonMonthDays is False):

•		1	May	2012	2		•
	Su	Мо	Tu	We	Th	Fr	Sa
18			1	2	3	4	5
19	6	7	8	9	10	11	12
20	13	14	15	16	17	18	19
21	20	21	22	23	24	25	26
22	27	28	29	30	31		
23							
		-	То	day)		

property Calendar.ShowTodayButton as Boolean

Retrieves or sets a value that indicates whether the today button is visible or hidden, in the calendar panel.

Туре	Description
Boolean	A Boolean expression that specifies whether the Today button is visible or hidden, in the calendar panel.

By default, the ShowTodayButton property is True. Use the ShowTodayButton property to hide the Today button. The <u>TodayCaption</u> property indicates the caption to be displayed on the Today's button. The <u>ShowYearScroll</u> property indicates whether the calendar panel displays an horizontal scroll bar to allow the user to change the calendar's year. The <u>Background</u>(exCalendarTodayForeColor) property indicates the Today's button foreground color.

- <u>Background</u>(exCalendarTodayUp) and <u>Background</u>(exCalendarTodayDown) properties indicate the visual appearance of the Today's button.
- <u>Background</u>(exCalendarMarkToday) and <u>Background</u>(exCalendarMarkTodayForeColor) properties indicates the background and foreground colors of the today date, in the calendar panel.
- <u>Background</u>(exScheduleMarkTodayBackColor) and <u>Background</u>(exScheduleMarkTodayForeColor) properties indicates the background and foreground colors of the today date, in the schedule panel.

property Calendar.ShowWeeks as Boolean

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

Туре	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 <u>SingleSel</u> is False, by clicking into the weeks header, the entire week is selected. The <u>DisplayWeekNumberAs</u> property specifies the way the control displays the week number. Use the <u>ShowNonMonthDays</u> 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 (in the calendar panel) to change the year is visible or hidden.

A Boolean expression that specifies whether the	
Boolean calendar's scroll is visible or hidden.	

By default, the ShowYearScroll property is False. The <u>Background</u>(exCalendarScrollRange) and <u>Background</u>(exCalendarScrollThumb) properties define the visual appearance of the scroll range and the thumb on the scroll. Use the <u>ShowTodayButton</u> property to hide the Today button. The <u>TodayCaption</u> property indicates the caption to be displayed on the Today's button.

property Calendar.SingleSel as Boolean

Returns or sets a value that indicates whether the user can select one or more dates in the calendar panel.

Туре	Description
Boolean	A Boolean expression that specifies whether the control

supports single or multiple selection.

By default, the SingleSel property is False, which means that the user can select multiple dates. The SelCount property counts the dates being selected in the calendar panel. The AllowSelectDate property indicates the keys combination so the user can select new dates in the calendar panel, and so, new dates to be shown in the schedule view. The AllowSelectDateRect specifies the keys combination so the user can do a rectangular selection in the calendar panel. Use the Selection/SelectDate property to change programmatically the dates being selected in the calendar, including the dates to be shown in the schedule view. You can use the Selection/SelCount/SelDate property to retrieve the selected dates. Once the user starts selecting a new date in the calendar panel, the control fires the LayoutStartChanging(exCalendarSelectionChange). Once a new date is selected, the LayoutEndChanging(exCalendarSelectionChange) event occurs. The DateFromPoint property indicates the date in the calendar panel from the cursor. The /NET and /WPF configurations provides the SelDates property equivalent of Selection, instead the SelDates returns a collection od DateTime objects (public virtual List<DateTime> SelDates). The <u>ClipToSel</u> property indicates whether the control clips the schedule panel to view the selected dates only.

The following samples show how you can enumerate the selected dates, in the calendar panel, once the <u>LayoutEndChanging</u>(exCalendarSelectionChange) event occurs.

VB

```
Private Sub Schedule1_LayoutEndChanging(ByVal Operation As
EXSCHEDULELibCtl.LayoutChangingEnum)
If Operation = exCalendarSelectionChange Then
Dim d As Variant
For Each d In Schedule1.Calendar.Selection
Debug.Print "Select: " & d
Next
End If
End Sub
```

VB/NET

Private Sub Exschedule1_LayoutEndChanging(ByVal sender As System.Object, ByVal Operation As exontrol.EXSCHEDULELib.LayoutChangingEnum) Handles Exschedule1.LayoutEndChanging If Operation = exontrol.EXSCHEDULELib.LayoutChangingEnum.exCalendarSelectionChange Then For Each d As DateTime In Exschedule1.Calendar.**SelDates** Debug.Print(d.ToString()) Next End If End Sub

or:

Private Sub Exschedule1_LayoutEndChanging(ByVal sender As System.Object, ByVal Operation As exontrol.EXSCHEDULELib.LayoutChangingEnum) Handles Exschedule1.LayoutEndChanging If Operation = exontrol.EXSCHEDULELib.LayoutChangingEnum.exCalendarSelectionChange Then For Each d As DateTime In Exschedule1.Calendar.**Selection** Debug.Print(d.ToString()) Next End If End Sub

C#

```
private void exschedule1_LayoutEndChanging(object sender,
exontrol.EXSCHEDULELib.LayoutChangingEnum Operation)
{
    if ( Operation ==
exontrol.EXSCHEDULELib.LayoutChangingEnum.exCalendarSelectionChange )
    {
      foreach (DateTime d in exschedule1.Calendar.SelDates)
        System.Diagnostics.Debug.Print(d.ToString());
    }
```

```
private void exschedule1_LayoutEndChanging(object sender,
exontrol.EXSCHEDULELib.LayoutChangingEnum Operation)
{
    if ( Operation ==
    exontrol.EXSCHEDULELib.LayoutChangingEnum.exCalendarSelectionChange )
    {
      foreach (DateTime d in exschedule1.Calendar.Selection as Array)
        System.Diagnostics.Debug.Print(d.ToString());
    }
}
```

VFP

```
*** ActiveX Control Event ***
LPARAMETERS operation
* 1 ' exCalendarSelectionChange
If Operation = 1 Then
for i = 0 to thisform.Schedule1.Calendar.SelCount() - 1
wait window TToC(thisform.Schedule1.Calendar.SelDate(i))
next
EndIf
```

C++

```
{
    VARIANT* pValue = (VARIANT*)p;
    if ( V_VT( pValue ) == VT_DATE )
    {
        CString strMessage;
        strMessage.Format(_T("Select: %f\r\n"), V_DATE( pValue ) );
        OutputDebugString( strMessage );
        }
    }
    SafeArrayUnaccessData( V_ARRAY( &selection ) );
    }
    }
}
```

where m_spSchedule is of EXSCHEDULELib::ISchedulePtr type.

property Calendar.TodayCaption as String

Retrieves or sets a value that indicates the today button's caption, in the calendar panel.

Туре	Description
String	A String expression that specifies the caption to be displayed on the Today's button

By default, the TodayCaption property is "Today". The TodayCaption property indicates the caption to be displayed on the Today's button. Use the <u>ShowTodayButton</u> property to hide the Today button. The <u>ShowYearScroll</u> property indicates whether the calendar panel displays an horizontal scroll bar to allow the user to change the calendar's year. The <u>Background</u>(exCalendarTodayForeColor) property indicates the Today's button foreground color. The <u>Background</u>(exCalendarTodayUp) and <u>Background</u>(exCalendarTodayDown) properties indicate the visual appearance of the Today's button.

property Calendar.WeekDays as String

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

Туре	Description
String	A String expression that indicates list of names for each week day, separated by space.

By default, the WeekDays property is "Sunday Monday Tuesday Wednesday Thursday Friday Saturday". Use the LocWeekDays property to get the locale list of week days as indicated by current regional settings. The <%d1%>, <%d2%>, <%d3%>, <%ddd%> or <%dddd%> HTML tags indicates the week day, as appropriate set by the WeekDays property. The <%loc_d1%>, <%loc_d2%>, <%loc_d3%>, <%loc_ddd%> or <%loc_dddd%> HTML tags indicates the week day, as appropriate set by the WeekDays property, using the current user regional and language settings (LocAMPM property). The FirstWeekDay property indicates the first day of the week. The MonthNames property specifies the list of name of the months. The AMPM property specifies the AM and PM indicators.

The following samples set the current view to display the locate date/time as set in the current regional settings.

VBA (MS Access, Excell...)

```
With Schedule1
With .Calendar
.FirstWeekDay = .LocFirstWeekDay
.MonthNames = .LocMonthNames
.WeekDays = .LocWeekDays
.AMPM = .LocAMPM
End With
End With
```

VB6

With Schedule1 With .Calendar .FirstWeekDay = .**LocFirstWeekDay** .MonthNames = .**LocMonthNames** .WeekDays = .**LocWeekDays**

```
.AMPM = .LocAMPM
End With
End With
```

VB.NET

With Exschedule1 With .Calendar .FirstWeekDay = .**LocFirstWeekDay** .MonthNames = .**LocMonthNames** .WeekDays = .**LocWeekDays** .AMPM = .**LocAMPM** End With End With

VB.NET for /COM

With AxSchedule1 With .Calendar .FirstWeekDay = .**LocFirstWeekDay** .MonthNames = .**LocMonthNames** .WeekDays = .**LocWeekDays** .AMPM = .**LocAMPM** End With End With

C++

```
/*
Copy and paste the following directives to your header file as
    it defines the namespace 'EXSCHEDULELib' for the library: 'ExSchedule 1.0 Control
Library'
#import <ExSchedule.dll>
    using namespace EXSCHEDULELib;
*/
EXSCHEDULELib::ISchedulePtr spSchedule1 = GetDlgItem(IDC_SCHEDULE1)-
>GetControlUnknown();
```

```
EXSCHEDULELib::ICalendarPtr var_Calendar = spSchedule1->GetCalendar();
var_Calendar->PutFirstWeekDay(var_Calendar->GetLocFirstWeekDay());
var_Calendar->PutMonthNames(var_Calendar->GetLocMonthNames());
var_Calendar->PutWeekDays(var_Calendar->GetLocWeekDays());
var_Calendar->PutAMPM(var_Calendar->GetLocAMPM());
```

C++ Builder

```
Exschedulelib_tlb::ICalendarPtr var_Calendar = Schedule1->Calendar;
var_Calendar->FirstWeekDay = var_Calendar->LocFirstWeekDay;
var_Calendar->MonthNames = var_Calendar->LocMonthNames;
var_Calendar->WeekDays = var_Calendar->LocWeekDays;
var_Calendar->AMPM = var_Calendar->LocAMPM;
```

C#

```
exontrol.EXSCHEDULELib.Calendar var_Calendar = exschedule1.Calendar;
var_Calendar.FirstWeekDay = var_Calendar.LocFirstWeekDay;
var_Calendar.MonthNames = var_Calendar.LocMonthNames;
var_Calendar.WeekDays = var_Calendar.LocWeekDays;
var_Calendar.AMPM = var_Calendar.LocAMPM;
```

JavaScript

```
<OBJECT classid="clsid:9B09E13D-7A88-4299-9DBE-383380435377"
id="Schedule1"></OBJECT>
<SCRIPT LANGUAGE="JScript">
var var_Calendar = Schedule1.Calendar;
var_Calendar.FirstWeekDay = var_Calendar.LocFirstWeekDay;
var_Calendar.MonthNames = var_Calendar.LocMonthNames;
var_Calendar.WeekDays = var_Calendar.LocWeekDays;
var_Calendar.AMPM = var_Calendar.LocAMPM;
</SCRIPT>
```

C# for /COM
```
EXSCHEDULELib.Calendar var_Calendar = axSchedule1.Calendar;
var_Calendar.FirstWeekDay = var_Calendar.LocFirstWeekDay;
var_Calendar.MonthNames = var_Calendar.LocMonthNames;
var_Calendar.WeekDays = var_Calendar.LocWeekDays;
var_Calendar.AMPM = var_Calendar.LocAMPM;
```

X++ (Dynamics Ax 2009)

```
public void init()
{
    COM com_Calendar;
    anytype var_Calendar;
    ;
    super();
    var_Calendar = exschedule1.Calendar(); com_Calendar = var_Calendar;
        com_Calendar.FirstWeekDay(com_Calendar.LocFirstWeekDay());
        com_Calendar.MonthNames(com_Calendar.LocMonthNames());
        com_Calendar.WeekDays(com_Calendar.LocWeekDays());
        com_Calendar.WeekDays(com_Calendar.LocWeekDays());
        com_Calendar.AMPM(com_Calendar.LocAMPM());
    }
}
```

Delphi 8 (.NET only)

```
with AxSchedule1 do
begin
with Calendar do
begin
FirstWeekDay := LocFirstWeekDay;
MonthNames := LocMonthNames;
WeekDays := LocWeekDays;
AMPM := LocAMPM;
end;
end
```

Delphi (standard)

```
with Schedule1 do
begin
with Calendar do
begin
FirstWeekDay := LocFirstWeekDay;
MonthNames := LocMonthNames;
WeekDays := LocWeekDays;
AMPM := LocAMPM;
end;
end
```

VFP

```
with thisform.Schedule1
with .Calendar
.FirstWeekDay = .LocFirstWeekDay
.MonthNames = .LocMonthNames
.WeekDays = .LocWeekDays
.AMPM = .LocAMPM
endwith
endwith
```

dBASE Plus

```
local oSchedule,var_Calendar
```

```
oSchedule = form.Activex1.nativeObject
var_Calendar = oSchedule.Calendar
var_Calendar.FirstWeekDay = var_Calendar.LocFirstWeekDay
var_Calendar.MonthNames = var_Calendar.LocMonthNames
var_Calendar.WeekDays = var_Calendar.LocWeekDays
var_Calendar.AMPM = var_Calendar.LocAMPM
```

XBasic (Alpha Five)

Dim oSchedule as P Dim var_Calendar as P

```
oSchedule = topparent:CONTROL_ACTIVEX1.activex
var_Calendar = oSchedule.Calendar
var_Calendar.FirstWeekDay = var_Calendar.LocFirstWeekDay
var_Calendar.MonthNames = var_Calendar.LocMonthNames
var_Calendar.WeekDays = var_Calendar.LocWeekDays
var_Calendar.AMPM = var_Calendar.LocAMPM
```

Visual Objects

local var_Calendar as ICalendar

var_Calendar := oDCOCX_Exontrol1:Calendar
var_Calendar:FirstWeekDay := var_Calendar:LocFirstWeekDay
var_Calendar:MonthNames := var_Calendar:LocMonthNames
var_Calendar:WeekDays := var_Calendar:LocWeekDays
var_Calendar:AMPM := var_Calendar:LocAMPM

PowerBuilder

OleObject oSchedule,var_Calendar oSchedule = ole_1.Object var_Calendar = oSchedule.Calendar var_Calendar.FirstWeekDay = var_Calendar.LocFirstWeekDay var_Calendar.MonthNames = var_Calendar.LocMonthNames var_Calendar.WeekDays = var_Calendar.LocWeekDays var_Calendar.MPM = var_Calendar.LocMPM

Event object

The Event object indicates an appointment in the schedule panel. The <u>Add</u> method adds programmatically a new event to the schedule. The <u>ShowEvents</u> property specifies the type of events the control display. The <u>ShowEventLabels</u> property indicates whether the events display ShortLabel, LongLabel / ExtraLabel properties of the event. The <u>ShowEventPictures</u> property shows or hides the event's pictures.

The Event object supports the following properties and method.

Name	Description
<u>AllDayEvent</u>	Indicates whether the event is an all day event.
BodyBackColor	Specifies the background color or the visual appearance of the event (body).
BodyBackgroundExt	Indicates additional colors, text, images that can be displayed on the event's background using the EBN string format.
BodyBackgroundExtValue	Specifies at runtime, the value of the giving property for specified part of the background extension.
BodyForeColor	Specifies the foreground color of the event (body).
<u>BodyPattern</u>	Specifies the pattern of the event (body).
Caption	Indicates the caption to be displayed on the event's label.
<u>ClearShowStatus</u>	Clears the status flag, so the ShowStatusEvent property indicates whether the current event shows or hides its status.
<u>ClearStatusColor</u>	Clears the status color flag, so the StatusEventColor property indicates the color to show the event's status.
<u>Client</u>	Returns the client area of the event.
<u>Editable</u>	Specifies whether the event's caption is editable.
End	Specifies the ending date/time of the event.
EndUpdateEvent	Adds programmatically updated properties of the calendar-event to undo/redo queue.
EnsureVisible	Scrolls the control to ensure that the current calendar- event fits the control's visible area.
ExtraLabel	Specifies the extra label to be displayed on the event.
ExtraLabelAlign	Indicates the alignment of the event's extra label.
	Specifies the list of extra pictures to be displayed on the

<u>ExtraPictures</u>	event.
ExtraPicturesAlign	Indicates the alignment of the event's extra picture.
<u>GroupID</u>	Specifies the identifier of the group where the Event object belongs.
<u>Handle</u>	Gets handle of the Event object.
<u>KnownProperty</u>	Specifies the value for the Event's property giving its identifier.
<u>LabelAlign</u>	Indicates the alignment of the event's long label.
LongLabel	Specifies the long label to be displayed on the event.
<u>MaxDate</u>	Indicates the max date for the event.
<u>MinDate</u>	Indicates the min date for the event.
Movable	Specifies whether the user can move the event.
<u>MoveBy</u>	Moves the event by specified time.
Pictures	Specifies the list of pictures to be displayed on the event.
PicturesAlign	Indicates the alignment of the event's picture.
<u>Repetitive</u>	Returns or sets the expression to determine the repetitive event.
Resizable	Specifies whether the user can resizes the event at runtime.
<u>Selectable</u>	Specifies whether the user can selects the event.
<u>Selected</u>	Selects or unselects the current event.
<u>ShortLabel</u>	Specifies the short label to be displayed on the event.
<u>ShowStatus</u>	Specifies whether the current event shows or hides its status.
<u>Start</u>	Specifies the starting date/time of the event.
<u>StartUpdateEvent</u>	Starts changing properties of the calendar-event, so EndUpdateEvent method adds programmatically updated properties to undo/redo queue.
<u>StatusColor</u>	Specifies the color of the event's status.
<u>StatusPattern</u>	Specifies the pattern of the event (status)
<u>ToolTip</u>	Indicates the tooltip to be shown when the cursor hovers the event.
<u>ToolTipTitle</u>	Indicates the title of the tooltip to be shown when the

cursor hovers the event.



Indicates any extra data associated with the Event object.

property Event.AllDayEvent as Boolean

Indicates whether the event is an all day event.

Туре	Description
Boolean	A boolean expression that indicates whether the event is an all-day event.

The AllDayEvent property indicates whether the time section of the event's margins are ignored. The <u>Start/End</u> properties indicates the event's margins. The Start and End properties may be identical, if the AllDayEvent property is True. The <u>AllowCreateAllDayEvent</u> property indicates whether the user can create all-day events at runtime. The user can create automatically All-Day events only, when the control displays no time scale (the schedule could be zoomed using <u>AllowResizeSchedule</u>, so no time scale is shown). Use the <u>ShowAllDayHeader</u> property to show the schedule's All-Day header so all All-Day evens are shown on this header. The <u>AllowAllDayEventScroll</u> property gets or sets a value that specifies whether the all-day event header supports scrolling. The <u>AllowSelectCreateEvent</u> property specifies whether the newly created event gets selected or highlighted.

The <u>KnownProperty</u>(exEventAllDay) property indicates the AllDayEvent property on a label property such as: <u>DefaultEventLongLabel</u>, <u>DefaultEventShortLabel</u>, <u>CreateEventLabel</u>, <u>UpdateEventsLabel</u>, <u>ShortLabel</u>, <u>LongLabel</u> and <u>ExtraLabel</u>. You can use the <u>KnownProperty</u>(exEventStartDate) property to extract the starting date of the event. You can use the <u>KnownProperty</u>(exEventEndDate) property to extract the ending date of the event.

property Event.BodyBackColor as Color

Specifies the background color or the visual appearance of the event (body).

Color A Color expression that specifies the background color to show the event's body. 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.	Туре	Description
	Color	A Color expression that specifies the background color to show the event's body. The last 7 bits in the high significant byte of the color to indicates the identifier of the skin being used. Use the <u>Add</u> 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.

By default, the BodyBackColor property is 0. The BodyBackColor property has effect only, if it is not-zero. The BodyBackColor property specifies the background color of the event's body. The <u>BodyEventBackColor</u> property specifies the background color to show the body for all events. The <u>EventBackColor</u> property specifies the event's background color if it belongs to a group. The <u>BodyForeColor</u> property specifies the foreground color to show the labels on the event. The <u>BodyPattern</u> property gives access to the pattern to be shown on the event's body. The <u>StatusColor</u> property indicates the color show the event's status. Use the <u>BodyBackgroundExt</u> property to apply multiple colors, texts, icons, images, patterns or frames to the event's background.

property Event.BodyBackgroundExt as String

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

Туре	Description
String	A String expression ("EBN String Format") that defines the layout of the UI to be applied on the object's background. The <u>syntax</u> 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 BodyBackgroundExt property is empty. Using the BodyBackgroundExt 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 <u>EBN</u> is a set of UI elements that are built as a tree where each element is anchored to its parent element. Use the <u>BodyBackgroundExtValue</u> property to change at runtime any UI property for any part that composes the EBN String Format. The BodyBackgroundExt property is applied right after setting the object's backcolor, and before drawing the default object's captions, icons or pictures.

Complex samples:





Easy samples:

• "[pattern=6]", shows the <u>BDiagonal</u> 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.

object	
	caption

• "[text=`flag`,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 <u>BDiagonal</u> pattern with a border arround on the lower-half part of the object's background.



 "root[text=`caption 2`,align=0x22](client[text=`caption 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 <u>eXButton</u> WYSWYG Builder helps you to generate or view the EBN String Format, in the **To String** field as shown in the following screen shot:

<u> EBN Builder (http</u>	o://www.exont	rol.com)			_ D _X
E ≥ E E C ×: Y: Y:	CX: 50	■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■	🔲 📄 🎦 art 1`.align=0x11		3
	11			• •	
Root Left _{10%} Bottom _{50%} Right _{50%}		.	part 1		
Left _{50%} Client Left _{50%}	4				
Right _{50%}	7			part 2	
Quit					

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

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

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

```
<digit> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
<decimal> ::= <digit> <decimal>
<hexadigit> ::= <digit> | "A" | "B" "C" | "D" | "E" "F"
<hexa> ::= <hexadigit><hexa>
<number> ::= <decimal> | "0x" <hexa>
<color> ::= <rqbcolor> | 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 BodyBackgroundExt 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: To String: 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)])



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



so, if we apply to our object we got:

object

property Event.BodyBackgroundExtValue(Index as IndexExtEnum, Property as BackgroundExtPropertyEnum) as Variant

Specifies at runtime, the value of the giving property for specified part of the background extension.



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	A <u>BackgroundExtPropertyEnum</u> expression that specifies
<u>BackgroundExtPropertyEnun</u>	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 BodyBackgroundExtValue property to change at runtime any UI property for any part that composes the EBN String Format. The BodyBackgroundExtValue property has no effect if the <u>BodyBackgroundExt</u> 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*

BodyBackgroundExt 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 BodyBackgroundExt to be the same for them, and next use the BodyBackgroundExtValue 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 BodyBackgroundExt on "bottom[50%,pattern=6,frame]"

we got:



so let's change the percent of 50% to 25% like BodyBackgroundExtValue(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:

```
.BodyBackgroundExt = "bottom[50%,pattern=6,frame]"
.BodyBackgroundExtValue(exIndexExt1, exClientExt) = "25%"
```

property Event.BodyForeColor as Color

Specifies the foreground color of the event (body).

Туре	Description
Color	A Color expression that specifies the event's foreground color.

By default, the BodyForeColor property is 0. The BodyForeColor property specifies the foreground color to show the labels on the event. The <u>BodyBackColor</u> property specifies the background color of the event's body. The <u>BodyEventForeColor</u> property specifies the foreground color to show the body for all events. The <u>EventForeColor</u> property specifies the event's foreground color if it belongs to a group. The <u>BodyPattern</u> property gives access to the pattern to be shown on the event's body. The <u>StatusColor</u> property indicates the color show the event's status.

property Event.BodyPattern as Pattern

Specifies the pattern of the event (body).

Туре	Description
Pattern	A Pattern object associated with the event's body.

By default, the BodyPattern.<u>Type</u> property exPatternEmpty which indicates that no pattern is shown, on the event's body. The <u>Color</u> property indicates the color to display the pattern. The <u>FrameColor</u> property indicates the color to show the frame, if the exPatternFrame flag is included in the Type property. The <u>EventPattern</u> property indicates the pattern to be shown when events belongs to different groups. The <u>StatusPattern</u> property specifies the pattern to be shown on the event's status.

property Event.Caption as String

Indicates the caption to be displayed on the event's label.

Туре	Description
String	A String expression that indicates the event's caption. The string expression in the Caption property does not support HTML tags. It is a plain text.

The Caption of the event/appointment is displayed only if the <%=%5%> tag is included in the any of the following label properties: <u>ShortLabel</u>, <u>LongLabel</u>, <u>ExtraLabel</u> or <u>ToolTip</u>. In case you need HTML tags to be displayed on the event's body, you should use the <u>LongLabel</u> and <u>ExtraLabel</u> properties of the Event object.

By default, the <u>ShortLabel</u> is displayed only when the event's client area is small. If enough space the <u>LongLabel</u> and <u>ExtraLabel</u> may be displayed separately. The ShortLabel can not display HTML tags, instead the ExtraLabel and LongLabel can. In conclusion, you can define arbitrary labels for any event, you can have an automated label to be displayed for each event with different results based on the <u>KnowProperty</u> values.

There are two samples on how you can use the Caption property like

- combined with the Label properties
- just the caption.

The following samples shows how to use the Caption property of the Event.

VBA (MS Access, Excell...)

```
With Schedule1
.DefaultEventLongLabel = "<%=%256%> <br> <b> <%=%5%>"
.DefaultEventShortLabel = .DefaultEventLongLabel
.Calendar.Selection = #6/20/2012#
With .Events
.Add(#6/20/2012 9:00:00 AM#,#6/20/2012 11:00:00 AM#).Caption = "caption
1"
With .Add(#6/20/2012 11:00:00 AM#,#6/20/2012 1:00:00 PM#)
.LongLabel = ""
.ExtraLabel = "<%=%5%>"
.Caption = "caption 2"
End With
With .Add(#6/20/2012 1:00:00 PM#,#6/20/2012 3:00:00 PM#)
```

```
.LongLabel = "<%=%256%> <br> <b> <%=%5%> <br> <fgcolor=808080> another text"
.Caption = "caption 3"
End With
End With
End With
```

VB6

```
With Schedule1
  .DefaultEventLongLabel = "<%=%256%><br><br><%=%5%>"
  .DefaultEventShortLabel = .DefaultEventLongLabel
  .Calendar.Selection = #6/20/2012#
  With Events
    .Add(#6/20/2012 9:00:00 AM#,#6/20/2012 11:00:00 AM#).Caption = "caption
1"
    With .Add(#6/20/2012 11:00:00 AM#,#6/20/2012 1:00:00 PM#)
      .LongLabel = ""
      .ExtraLabel = "<%=%5%>"
      .Caption = "caption 2"
    End With
    With .Add(#6/20/2012 1:00:00 PM#,#6/20/2012 3:00:00 PM#)
      .LongLabel = "<%=%256%> <br> <b> <%=%5%> <br>
<fgcolor=808080>another text"
      .Caption = "caption 3"
    End With
  End With
End With
```

VB.NET

```
With Exschedule1
.DefaultEventLongLabel = "<%=%256%> <br> <b> <%=%5%>"
.DefaultEventShortLabel = .DefaultEventLongLabel
.Calendar.Selection = #6/20/2012#
With .Events
.Add(#6/20/2012 9:00:00 AM#,#6/20/2012 11:00:00 AM#).Caption = "caption
1"
```

```
With .Add(#6/20/2012 11:00:00 AM#,#6/20/2012 1:00:00 PM#)

.LongLabel = ""

.ExtraLabel = "<%=%5%>"

.Caption = "caption 2"

End With

With .Add(#6/20/2012 1:00:00 PM#,#6/20/2012 3:00:00 PM#)

.LongLabel = "<%=%256%> <br> <b> <%=%5%> <br>

<fgcolor=808080> another text"

.Caption = "caption 3"

End With

End With

End With

End With
```

VB.NET for /COM

```
With AxSchedule1
  .DefaultEventLongLabel = "<%=%256%><br><br><%=%5%>"
  .DefaultEventShortLabel = .DefaultEventLongLabel
  .Calendar.Selection = #6/20/2012#
  With Events
    .Add(#6/20/2012 9:00:00 AM#,#6/20/2012 11:00:00 AM#).Caption = "caption
1"
    With .Add(#6/20/2012 11:00:00 AM#,#6/20/2012 1:00:00 PM#)
      .LongLabel = ""
      .ExtraLabel = "<%=%5%>"
      .Caption = "caption 2"
    End With
    With .Add(#6/20/2012 1:00:00 PM#,#6/20/2012 3:00:00 PM#)
      .LongLabel = "<%=%256%> <br> <b> <%=%5%> <br>
<fgcolor=808080>another text"
      .Caption = "caption 3"
    End With
  End With
End With
```

C++

```
Copy and paste the following directives to your header file as
it defines the namespace 'EXSCHEDULELib' for the library: 'ExSchedule 1.0 Control
Library'
```

```
#import <ExSchedule.dll>
     using namespace EXSCHEDULELib;
*/
EXSCHEDULELib::ISchedulePtr spSchedule1 = GetDlgItem(IDC_SCHEDULE1)-
>GetControlUnknown();
spSchedule1->PutDefaultEventLongLabel(L"<%=%256%><br><br><b?(L"<%=%256%><br><b?(L"<%=%256%)<br><b?(L"<%=%256%)<br><b?(L"<%=%256%)<br><b?(L"<%=%256%)<br><b?(L"<%=%256%)<br><b?(L"<%=%256%)<br><b?(L"<%=%256%)<br><b?(L"<%=%256%)<br><b?(L"<%=%256%)<br><b?(L"<%=%256%)<br><b?(L"<%=%256%)<br><b?(L"<%=%256%)<br><b?(L"<%=%256%)<br><b?(L"<%=%256%)<br><b?(L"<%=%256%)<br><b?(L"<%=%256%)<br><b?(L"<%=%256%)<br><b?(L"<%=%256%)<br><b?(L"<%=%256%)<br/><br/>><br/>(L"<%=%256%)<br/><br/>(L"<%=%256%)<br/>(L"<%=%256%)<br/>(L"<%=%256%)<br/>(L"<%=%256%)<br/>(L"<%=%256%)<br/>(L"<%=%256%)<br/>(L"<%=%256%)<br/>(L"<%=%256%)<br/>(L"<%=%256%)<br/>(L"<%=%256%)<br/>(L"<%=%256%)<br/>(L"<%=%256%)<br/>(L"<%=%256%)<br/>(L"<%=%256%)<br/>(L"<%=%256%)<br/>(L"<%=%256%)<br/>(L"<%=%256%)<br/>(L"<%=%256%)<br/>(L"<%=%256%)<br/>(L"<%=%256%)<br/>(L"<%=%256%)<br/>(L"<%=%256%)<br/>(L"<%=%256%)<br/>(L"<%=%256%)<br/>(L"<%=%26%)<br/>(L"<%>%26%)<br/>(L"<%=%26%)<br/>(L"<%=%26%)<br/>(L"<%=%26%)<br/>(L"<%=%26%)<br/>(L"<%=%26%)<br/>(L"<%=%26%)<br/>(L"<%=%26%)<br/>(L"<%=%26%)<br/>(L"<%=%26%)<br/>(L"<%=%26%)<br/>(L"<%=%26%)<br/>(L"<%=%26%)<br/>(L"<%=%26%)<br/>(L"<%=%26%)<br/>(L"<%=%26%)<br/>(L"<%=%26%)<br/>(L"<%=%26%)<br/>(L"<%=%26%)<br/>(L"<%=%26%)<br/>(L"<%=%26%)<br/>(L"<%=%26%)<br/>(L"<%=%26%)<br/>(L"<%=%26%)<br/>(L"<%=%26%)<br/>(L"<%=%26%)<br/>(L"<%26%)<br/>(L"<%26%)<br/>(L"<%26%)<br/>(L"<%26%)<br/>(L"<%26%)<br/>(L"<%26%)<br/>(L"<%26%)<br/>(L"<%26%)<br/>(L"<%26%)<br/>(L"<%26%)<br/>(L"<%26%)<br/>(L"<%26%)<br/>(L"<%26%)<br/>(L"<%26%)<br/>(L"<%26%)<br/>(L"<%26%)<br/>(L"<%26%)<br/>(L"<%26%)<br/>(L"<%26%)<br/>(L"<%26%)<br/>(L"<%26%)<br/>(L"<%26%)<br/>(L"<%26%)<br/>(L"<%26%)<br/>(L"<%26%)<br/>(L"<%26%)<br/>(L"<%26%)<br/>(L"<%26%)<br/>(L"<%26%)<br/>(L"<%26%)<br/>(L"<%26%)<br/>(L"<%26%)<br/>(L"<%26%)<br/>(L"<%26%)<br/>(L"<%26%)<br/>(L"<%26%)<br/>(L"<%26%)<br/>(L"<%26%)<br/>(L"<%26%)<br/>(L"
spSchedule1->PutDefaultEventShortLabel(spSchedule1-
>GetDefaultEventLongLabel());
spSchedule1->GetCalendar()->PutSelection("6/20/2012");
EXSCHEDULELib::IEventsPtr var_Events = spSchedule1->GetEvents();
     var_Events->Add("6/20/2012 9:00:00 AM","6/20/2012 11:00:00 AM")-
> PutCaption(L"caption 1");
     EXSCHEDULELib::IEventPtr var_Event = var_Events->Add("6/20/2012 11:00:00
AM","6/20/2012 1:00:00 PM");
           var_Event->PutLongLabel(L"");
           var_Event->PutExtraLabel(L"<%=%5%>");
           var_Event->PutCaption(L"caption 2");
     EXSCHEDULELib::IEventPtr var_Event1 = var_Events->Add("6/20/2012 1:00:00
PM","6/20/2012 3:00:00 PM");
           <fgcolor=808080>another text");
           var_Event1->PutCaption(L"caption 3");
```

C++ Builder

```
Schedule1-> DefaultEventLongLabel = L" <%=%256%> <br> <b> <%=%5%>";
Schedule1-> DefaultEventShortLabel = Schedule1-> DefaultEventLongLabel;
Schedule1-> Calendar-> set_Selection(TVariant(TDateTime(2012,6,20).operator
double()));
Exschedulelib_tlb::IEventsPtr var_Events = Schedule1-> Events;
var_Events-> Add(TVariant(TDateTime(2012,6,20,9,00,00,0).operator
double()),TVariant(TDateTime(2012,6,20,11,00,00,0).operator double()))-> Caption =
```

```
L"caption 1";

Exschedulelib_tlb::IEventPtr var_Event = var_Events-

> Add(TVariant(TDateTime(2012,6,20,11,00,00,0).operator double()));

var_Event-> LongLabel = L"";

var_Event-> ExtraLabel = L"<%=%5%>";

var_Event-> Caption = L"caption 2";

Exschedulelib_tlb::IEventPtr var_Event1 = var_Events-

> Add(TVariant(TDateTime(2012,6,20,13,00,00,0).operator double()));

var_Event1-> LongLabel = L"<%=%256%> <br> <b> <%=%5%> <br> <fgcolor=808080> another text";

var_Event1-> Caption = L"caption 3";
```

C#

```
exschedule1.DefaultEventShortLabel = exschedule1.DefaultEventLongLabel;
exschedule1.Calendar.Selection =
Convert.ToDateTime("6/20/2012",System.Globalization.CultureInfo.GetCultureInfo("en-
US"));
exontrol.EXSCHEDULELib.Events var_Events = exschedule1.Events;
  var_Events.Add(Convert.ToDateTime("6/20/2012 9:00:00
AM", System. Globalization. CultureInfo. GetCultureInfo("en-
US")),Convert.ToDateTime("6/20/2012 11:00:00
AM",System.Globalization.CultureInfo.GetCultureInfo("en-US"))).Caption = "caption
1";
  exontrol.EXSCHEDULELib.Event var Event =
var_Events.Add(Convert.ToDateTime("6/20/2012 11:00:00
AM", System. Globalization. CultureInfo. GetCultureInfo("en-
US")),Convert.ToDateTime("6/20/2012 1:00:00
PM",System.Globalization.CultureInfo.GetCultureInfo("en-US")));
    var_Event.LongLabel = "";
    var Event.ExtraLabel = "<%=%5%>";
    var_Event.Caption = "caption 2";
  exontrol.EXSCHEDULELib.Event var_Event1 =
```

```
var_Events.Add(Convert.ToDateTime("6/20/2012 1:00:00
PM",System.Globalization.CultureInfo.GetCultureInfo("en-
US")),Convert.ToDateTime("6/20/2012 3:00:00
PM",System.Globalization.CultureInfo.GetCultureInfo("en-US")));
    var_Event1.LongLabel = "<%=%256%> <br> <b> <%=%5%> <br> <fgcolor=808080>another text";
    var_Event1.Caption = "caption 3";
```

JavaScript

```
<OBJECT classid = "clsid:9B09E13D-7A88-4299-9DBE-383380435377"
id="Schedule1"></OBJECT>
<SCRIPT LANGUAGE="JScript">
  Schedule1.DefaultEventLongLabel = "<%=%256%> <br> <br> <br> > ';
  Schedule1.DefaultEventShortLabel = Schedule1.DefaultEventLongLabel;
  Schedule1.Calendar.Selection = "6/20/2012";
  var var_Events = Schedule1.Events;
    var_Events.Add("6/20/2012 9:00:00 AM","6/20/2012 11:00:00 AM").Caption =
"caption 1";
    var var_Event = var_Events.Add("6/20/2012 11:00:00 AM","6/20/2012 1:00:00
PM");
      var_Event.LongLabel = "";
      var_Event.ExtraLabel = "<%=%5%>";
      var_Event.Caption = "caption 2";
    var var_Event1 = var_Events.Add("6/20/2012 1:00:00 PM","6/20/2012 3:00:00
PM");
      var_Event1.LongLabel = "<%=%256%> <br> <b> <%=%5%> <br>
<fgcolor=808080>another text";
      var_Event1.Caption = "caption 3";
</SCRIPT>
```

C# for /COM

axSchedule1.**DefaultEventLongLabel** = "<%=%256%>
 <%=%5%>"; axSchedule1.DefaultEventShortLabel = axSchedule1.**DefaultEventLongLabel**; axSchedule1.Calendar.Selection =

```
Convert.ToDateTime("6/20/2012",System.Globalization.CultureInfo.GetCultureInfo("en-
US"));
EXSCHEDULELib.Events var_Events = axSchedule1.Events;
  var_Events.Add(Convert.ToDateTime("6/20/2012 9:00:00
AM", System. Globalization. CultureInfo. GetCultureInfo("en-
US")),Convert.ToDateTime("6/20/2012 11:00:00
AM", System. Globalization. CultureInfo. GetCultureInfo("en-US"))). Caption = "caption
1":
  EXSCHEDULELib.Event var_Event =
var_Events.Add(Convert.ToDateTime("6/20/2012 11:00:00
AM", System. Globalization. CultureInfo. GetCultureInfo("en-
US")),Convert.ToDateTime("6/20/2012 1:00:00
PM",System.Globalization.CultureInfo.GetCultureInfo("en-US")));
    var_Event.LongLabel = "";
    var_Event.ExtraLabel = "<%=%5%>";
    var_Event.Caption = "caption 2";
  EXSCHEDULELib.Event var Event1 =
var_Events.Add(Convert.ToDateTime("6/20/2012 1:00:00
PM",System.Globalization.CultureInfo.GetCultureInfo("en-
US")),Convert.ToDateTime("6/20/2012 3:00:00
PM",System.Globalization.CultureInfo.GetCultureInfo("en-US")));
    var_Event1.LongLabel = "<%=%256%> <br> <b> <%=%5%> <br>
<fgcolor=808080>another text";
    var_Event1.Caption = "caption 3";
```

X++ (Dynamics Ax 2009)

```
public void init()
{
    COM com_Event,com_Event1,com_Events;
    anytype var_Event,var_Event1,var_Events;
    ;
    super();
    exschedule1.DefaultEventLongLabel("<%=%256%> <br> <b> <%=%5%>");
```

```
exschedule1.DefaultEventShortLabel(exschedule1.DefaultEventLongLabel());
```

```
exschedule1.Calendar().Selection(COMVariant::createFromDate(str2Date("6/20/2012",2
  var_Events = exschedule1.Events(); com_Events = var_Events;
   var Event =
COM::createFromObject(com_Events.Add(COMVariant::createFromUtcDateTime(str2Da
9:00:00",213)),COMVariant::createFromUtcDateTime(str2Datetime("6/20/2012
11:00:00",213)))); com_Event = var_Event;
    com_Event.Caption("caption 1");
   var Event =
com_Events.Add(COMVariant::createFromUtcDateTime(str2Datetime("6/20/2012
11:00:00",213)),COMVariant::createFromUtcDateTime(str2Datetime("6/20/2012
13:00:00",213))); com_Event = var_Event;
     com_Event.LongLabel("");
     com_Event.ExtraLabel("<%=%5%>");
      com_Event.Caption("caption 2");
   var Event1 =
com_Events.Add(COMVariant::createFromUtcDateTime(str2Datetime("6/20/2012
13:00:00",213)),COMVariant::createFromUtcDateTime(str2Datetime("6/20/2012
15:00:00",213))); com_Event1 = var_Event1;
      <fgcolor=808080>another text");
      com_Event1.Caption("caption 3");
```

Delphi 8 (.NET only)

```
with AxSchedule1 do
begin
    DefaultEventLongLabel := '<%=%256%> <br> <b> <%=%5%>';
    DefaultEventShortLabel := DefaultEventLongLabel;
    Calendar.Selection := '6/20/2012';
    with Events do
    begin
        Add('6/20/2012 9:00:00 AM','6/20/2012 11:00:00 AM').Caption := 'caption 1';
        with Add('6/20/2012 11:00:00 AM','6/20/2012 1:00:00 PM') do
```

```
begin
   LongLabel := '';
   ExtraLabel := '<%=%5%>';
   Caption := 'caption 2';
   end;
   with Add('6/20/2012 1:00:00 PM','6/20/2012 3:00:00 PM') do
   begin
   LongLabel := '<%=%256%> <br> <b> <%=%5%> <br> <fgcolor=808080> another text';
   Caption := 'caption 3';
   end;
   end;
end;
```

Delphi (standard)

```
with Schedule1 do
begin
  DefaultEventLongLabel := '<%=%256%> <br> <b> <%=%5%>';
  DefaultEventShortLabel := DefaultEventLongLabel;
  Calendar.Selection := '6/20/2012';
  with Events do
  begin
    Add('6/20/2012 9:00:00 AM','6/20/2012 11:00:00 AM').Caption := 'caption 1';
    with Add('6/20/2012 11:00:00 AM','6/20/2012 1:00:00 PM') do
    begin
      LongLabel := ";
      ExtraLabel := '<%=%5%>';
      Caption := 'caption 2';
    end;
    with Add('6/20/2012 1:00:00 PM','6/20/2012 3:00:00 PM') do
    begin
      LongLabel := '<%=%256%> <br> <b> <%=%5%> <br>
<fgcolor=808080>another text';
      Caption := 'caption 3';
    end;
  end;
```

end

VFP

```
with thisform.Schedule1
  .DefaultEventLongLabel = "<%=%256%><br><br><%=%5%>"
  .DefaultEventShortLabel = .DefaultEventLongLabel
  .Calendar.Selection = ^{2012-6-20}
  with .Events
    .Add({^2012-6-20 9:00:00},{^2012-6-20 11:00:00}).Caption = "caption 1"
    with .Add({^2012-6-20 11:00:00},{^2012-6-20 13:00:00})
      .LongLabel = ""
      .ExtraLabel = "<%=%5%>"
      .Caption = "caption 2"
    endwith
    with .Add({^2012-6-20 13:00:00},{^2012-6-20 15:00:00})
      .LongLabel = "<%=%256%> <br> <b> <%=%5%> <br>
<fgcolor=808080>another text"
      .Caption = "caption 3"
    endwith
  endwith
endwith
```

dBASE Plus

```
local oSchedule,var_Event,var_Event1,var_Event2,var_Events
oSchedule = form.Activex1.nativeObject
oSchedule.DefaultEventLongLabel = "<%=%256%> <br> <b> <%=%5%>"
oSchedule.DefaultEventShortLabel = oSchedule.DefaultEventLongLabel
oSchedule.Calendar.Selection = "06/20/2012"
var_Events = oSchedule.Events
// var_Events.Add("06/20/2012 09:00:00","06/20/2012 11:00:00").Caption
= "caption 1"
var_Event = var_Events.Add("06/20/2012 09:00:00","06/20/2012 11:00:00")
with (oSchedule)
TemplateDef = [Dim var_Event]
TemplateDef = var_Event
```

```
Template = [var_Event.Caption = "caption 1"]
endwith
var_Event1 = var_Events.Add("06/20/2012 11:00:00","06/20/2012 13:00:00")
var_Event1.LongLabel = ""
var_Event1.ExtraLabel = "<%=%5%>"
var_Event1.Caption = "caption 2"
var_Event2 = var_Events.Add("06/20/2012 13:00:00","06/20/2012 15:00:00")
var_Event2.LongLabel = "<%=%256%> <br> <b> <%=%5%> <br> <
fgcolor=808080> another text"
var_Event2.Caption = "caption 3"
```

XBasic (Alpha Five)

```
Dim oSchedule as P
Dim var Event as P
Dim var Event1 as P
Dim var Event2 as P
Dim var Events as P
oSchedule = topparent:CONTROL_ACTIVEX1.activex
oSchedule.DefaultEventLongLabel = "<%=%256%> <br> <br> <br> > <%=%5%>"
oSchedule.DefaultEventShortLabel = oSchedule.DefaultEventLongLabel
oSchedule.Calendar.Selection = {06/20/2012}
var Events = oSchedule.Events
  'var_Events.Add({06/20/2012 09:00:00},{06/20/2012 11:00:00}).Caption =
"caption 1"
  var_Event = var_Events.Add({06/20/2012 09:00:00},{06/20/2012 11:00:00})
  oSchedule.TemplateDef = "Dim var_Event"
  oSchedule.TemplateDef = var_Event
  oSchedule.Template = "var_Event.Caption = \"caption 1\""
  var_Event1 = var_Events.Add(\{06/20/2012 \ 11:00:00\}, \{06/20/2012 \ 13:00:00\})
    var_Event1.LongLabel = ""
    var Event1.ExtraLabel = "<%=%5%>"
    var_Event1.Caption = "caption 2"
  var_Event2 = var_Events.Add({06/20/2012 13:00:00},{06/20/2012 15:00:00})
```

```
var_Event2.LongLabel = "<%=%256%> <br> <b> <%=%5%> <br> <fgcolor=808080> another text"
  var_Event2.Caption = "caption 3"
```

Visual Objects

```
local var_Event, var_Event1 as IEvent
local var Events as IEvents
oDCOCX_Exontrol1:DefaultEventLongLabel := "<%=%256%> <br> <b>
<%=%5%>"
oDCOCX_Exontrol1:DefaultEventShortLabel :=
oDCOCX_Exontrol1:DefaultEventLongLabel
oDCOCX_Exontrol1:Calendar:Selection := SToD("20120620")
var_Events := oDCOCX_Exontrol1:Events
  var_Events:Add(SToD("20120620 09:00:00"),SToD("20120620 11:00:00")):Caption
:= "caption 1"
  var_Event := var_Events:Add(SToD("20120620 11:00:00"),SToD("20120620
13:00:00"))
    var_Event:LongLabel := ""
    var Event:ExtraLabel := "<%=%5%>"
    var_Event:Caption := "caption 2"
  var_Event1 := var_Events:Add(SToD("20120620 13:00:00"),SToD("20120620
15:00:00"))
    var_Event1:LongLabel := "<%=%256%><br><b><%=%5%><br>
<fgcolor=808080>another text"
    var_Event1:Caption := "caption 3"
```

PowerBuilder

OleObject oSchedule,var_Event,var_Event1,var_Events

oSchedule = ole_1.Object oSchedule.**DefaultEventLongLabel** = "<%=%256%>
 <%=%5%>" oSchedule.DefaultEventShortLabel = oSchedule.**DefaultEventLongLabel** oSchedule.Calendar.Selection = 2012-06-20

```
var_Events = oSchedule.Events
var_Events.Add(DateTime(2012-06-20,09:00:00),DateTime(2012-06-
20,11:00:00)).Caption = "caption 1"
var_Event = var_Events.Add(DateTime(2012-06-20,11:00:00),DateTime(2012-06-
20,13:00:00))
var_Event.LongLabel = ""
var_Event.ExtraLabel = "<%=%5%>"
var_Event.Caption = "caption 2"
var_Event1 = var_Events.Add(DateTime(2012-06-20,13:00:00),DateTime(2012-06-
20,15:00:00))
var_Event1.LongLabel = "<%=%256%> <br> <b> <%=%5%> <br> <
<fgcolor=808080> another text"
var_Event1.Caption = "caption 3"
```

The following sample uses and display just the Caption of the event.

VBA (MS Access, Excell...)

```
With Schedule1
.DefaultEventLongLabel = "<%=%5%>"
.DefaultEventShortLabel = .DefaultEventLongLabel
.Calendar.Selection = #6/20/2012#
With .Events
.Add(#6/20/2012 9:00:00 AM#,#6/20/2012 11:00:00 AM#).Caption = "caption
1"
.Add(#6/20/2012 11:00:00 AM#,#6/20/2012 1:00:00 PM#).Caption = "caption
2"
.Add(#6/20/2012 1:00:00 PM#,#6/20/2012 3:00:00 PM#).Caption = "caption 3"
End With
End With
```

VB6

```
With Schedule1

.DefaultEventLongLabel = "<%=%5%>"

.DefaultEventShortLabel = .DefaultEventLongLabel

.Calendar.Selection = #6/20/2012#
```

```
With .Events

.Add(#6/20/2012 9:00:00 AM#,#6/20/2012 11:00:00 AM#).Caption = "caption

1"

.Add(#6/20/2012 11:00:00 AM#,#6/20/2012 1:00:00 PM#).Caption = "caption

2"

.Add(#6/20/2012 1:00:00 PM#,#6/20/2012 3:00:00 PM#).Caption = "caption 3"

End With

End With
```

VB.NET

```
With Exschedule1
.DefaultEventLongLabel = "<%=%5%>"
.DefaultEventShortLabel = .DefaultEventLongLabel
.Calendar.Selection = #6/20/2012#
With .Events
.Add(#6/20/2012 9:00:00 AM#,#6/20/2012 11:00:00 AM#).Caption = "caption
1"
.Add(#6/20/2012 11:00:00 AM#,#6/20/2012 1:00:00 PM#).Caption = "caption
2"
.Add(#6/20/2012 1:00:00 PM#,#6/20/2012 3:00:00 PM#).Caption = "caption 3"
End With
End With
```

VB.NET for /COM

```
With AxSchedule1

.DefaultEventLongLabel = "<%=%5%>"

.DefaultEventShortLabel = .DefaultEventLongLabel

.Calendar.Selection = #6/20/2012#

With .Events

.Add(#6/20/2012 9:00:00 AM#,#6/20/2012 11:00:00 AM#).Caption = "caption

1"

.Add(#6/20/2012 11:00:00 AM#,#6/20/2012 1:00:00 PM#).Caption = "caption

2"

.Add(#6/20/2012 1:00:00 PM#,#6/20/2012 3:00:00 PM#).Caption = "caption 3"

End With

End With
```

```
C++
```

```
Copy and paste the following directives to your header file as
  it defines the namespace 'EXSCHEDULELib' for the library: 'ExSchedule 1.0 Control
Library'
  #import <ExSchedule.dll>
  using namespace EXSCHEDULELib;
*/
EXSCHEDULELib::ISchedulePtr spSchedule1 = GetDlgItem(IDC_SCHEDULE1)-
>GetControlUnknown();
spSchedule1->PutDefaultEventLongLabel(L"<%=%5%>");
spSchedule1->PutDefaultEventShortLabel(spSchedule1-
>GetDefaultEventLongLabel());
spSchedule1->GetCalendar()->PutSelection("6/20/2012");
EXSCHEDULELib::IEventsPtr var_Events = spSchedule1->GetEvents();
  var_Events->Add("6/20/2012 9:00:00 AM","6/20/2012 11:00:00 AM")-
> PutCaption(L"caption 1");
  var_Events->Add("6/20/2012 11:00:00 AM","6/20/2012 1:00:00 PM")-
> PutCaption(L"caption 2");
  var_Events->Add("6/20/2012 1:00:00 PM","6/20/2012 3:00:00 PM")-
> PutCaption(L"caption 3");
```

C++ Builder

```
Schedule1->DefaultEventLongLabel = L"<%=%5%>";
Schedule1->DefaultEventShortLabel = Schedule1->DefaultEventLongLabel;
Schedule1->Calendar->set_Selection(TVariant(TDateTime(2012,6,20).operator
double()));
Exschedulelib_tlb::IEventsPtr var_Events = Schedule1->Events;
var_Events->Add(TVariant(TDateTime(2012,6,20,9,00,00,0).operator
double()),TVariant(TDateTime(2012,6,20,11,00,00,0).operator double()))->Caption =
L"caption 1";
var_Events->Add(TVariant(TDateTime(2012,6,20,11,00,00,0).operator
double()),TVariant(TDateTime(2012,6,20,13,00,00,0).operator double()))->Caption =
L"caption 2";
```

```
var_Events->Add(TVariant(TDateTime(2012,6,20,13,00,00,0).operator
double()),TVariant(TDateTime(2012,6,20,15,00,00,0).operator double()))->Caption =
L"caption 3";
```

C#

```
exschedule1.DefaultEventLongLabel = "<%=%5%>";
exschedule1.DefaultEventShortLabel = exschedule1.DefaultEventLongLabel;
exschedule1.Calendar.Selection =
Convert.ToDateTime("6/20/2012",System.Globalization.CultureInfo.GetCultureInfo("en-
US"));
exontrol.EXSCHEDULELib.Events var_Events = exschedule1.Events;
  var_Events.Add(Convert.ToDateTime("6/20/2012 9:00:00
AM", System. Globalization. CultureInfo. GetCultureInfo("en-
US")),Convert.ToDateTime("6/20/2012 11:00:00
AM", System. Globalization. CultureInfo. GetCultureInfo("en-US"))). Caption = "caption
1";
  var_Events.Add(Convert.ToDateTime("6/20/2012 11:00:00
AM", System. Globalization. CultureInfo. GetCultureInfo("en-
US")),Convert.ToDateTime("6/20/2012 1:00:00
PM",System.Globalization.CultureInfo.GetCultureInfo("en-US"))).Caption = "caption
2";
  var_Events.Add(Convert.ToDateTime("6/20/2012 1:00:00
PM",System.Globalization.CultureInfo.GetCultureInfo("en-
US")),Convert.ToDateTime("6/20/2012 3:00:00
PM",System.Globalization.CultureInfo.GetCultureInfo("en-US"))).Caption = "caption
3";
```

JavaScript

```
<OBJECT classid="clsid:9B09E13D-7A88-4299-9DBE-383380435377"
id="Schedule1"></OBJECT>
```

<SCRIPT LANGUAGE="JScript">

Schedule1.**DefaultEventLongLabel** = "<%=%5%>"; Schedule1.**DefaultEventShortLabel** = Schedule1.**DefaultEventLongLabel**;
```
Schedule1.Calendar.Selection = "6/20/2012";
var var_Events = Schedule1.Events;
var_Events.Add("6/20/2012 9:00:00 AM","6/20/2012 11:00:00 AM").Caption =
"caption 1";
var_Events.Add("6/20/2012 11:00:00 AM","6/20/2012 1:00:00 PM").Caption =
"caption 2";
var_Events.Add("6/20/2012 1:00:00 PM","6/20/2012 3:00:00 PM").Caption =
"caption 3";
</SCRIPT>
```

C# for /COM

```
axSchedule1.DefaultEventLongLabel = "<%=%5%>";
axSchedule1.DefaultEventShortLabel = axSchedule1.DefaultEventLongLabel;
axSchedule1.Calendar.Selection =
Convert.ToDateTime("6/20/2012",System.Globalization.CultureInfo.GetCultureInfo("en-
US"));
EXSCHEDULELib.Events var_Events = axSchedule1.Events;
  var_Events.Add(Convert.ToDateTime("6/20/2012 9:00:00
AM", System. Globalization. CultureInfo. GetCultureInfo("en-
US")),Convert.ToDateTime("6/20/2012 11:00:00
AM", System. Globalization. CultureInfo. GetCultureInfo("en-US"))). Caption = "caption
1";
  var_Events.Add(Convert.ToDateTime("6/20/2012 11:00:00
AM", System. Globalization. CultureInfo. GetCultureInfo("en-
US")),Convert.ToDateTime("6/20/2012 1:00:00
PM",System.Globalization.CultureInfo.GetCultureInfo("en-US"))).Caption = "caption
2";
  var_Events.Add(Convert.ToDateTime("6/20/2012 1:00:00
PM",System.Globalization.CultureInfo.GetCultureInfo("en-
US")),Convert.ToDateTime("6/20/2012 3:00:00
PM",System.Globalization.CultureInfo.GetCultureInfo("en-US"))).Caption = "caption
3";
```

X++ (Dynamics Ax 2009)

public void init()

```
COM com_Event,com_Events;
anytype var_Event,var_Events;
```

```
super();
```

```
exschedule1.DefaultEventLongLabel("<%=%5%>");
exschedule1.DefaultEventShortLabel(exschedule1.DefaultEventLongLabel());
```

```
exschedule1.Calendar().Selection(COMVariant::createFromDate(str2Date("6/20/2012",2
```

```
var_Events = exschedule1.Events(); com_Events = var_Events;
    var_Event =
COM::createFromObject(com_Events.Add(COMVariant::createFromUtcDateTime(str2Da
9:00:00",213)),COMVariant::createFromUtcDateTime(str2Datetime("6/20/2012
11:00:00",213)))); com_Event = var_Event;
    com_Event.Caption("caption 1");
    var Event =
COM::createFromObject(com_Events.Add(COMVariant::createFromUtcDateTime(str2Da
11:00:00",213)),COMVariant::createFromUtcDateTime(str2Datetime("6/20/2012
13:00:00",213)))); com_Event = var_Event;
    com_Event.Caption("caption 2");
    var Event =
COM::createFromObject(com_Events.Add(COMVariant::createFromUtcDateTime(str2Da
13:00:00",213)),COMVariant::createFromUtcDateTime(str2Datetime("6/20/2012
15:00:00",213)))); com_Event = var_Event;
    com_Event.Caption("caption 3");
```

Delphi 8 (.NET only)

```
with AxSchedule1 do
begin
    DefaultEventLongLabel := '<%=%5%>';
    DefaultEventShortLabel := DefaultEventLongLabel;
    Calendar.Selection := '6/20/2012';
```

```
with Events do
begin
    Add('6/20/2012 9:00:00 AM','6/20/2012 11:00:00 AM').Caption := 'caption 1';
    Add('6/20/2012 11:00:00 AM','6/20/2012 1:00:00 PM').Caption := 'caption 2';
    Add('6/20/2012 1:00:00 PM','6/20/2012 3:00:00 PM').Caption := 'caption 3';
    end;
end
```

Delphi (standard)

```
with Schedule1 do
begin
    DefaultEventLongLabel := '<%=%5%>';
    DefaultEventShortLabel := DefaultEventLongLabel;
    Calendar.Selection := '6/20/2012';
    with Events do
    begin
        Add('6/20/2012 9:00:00 AM','6/20/2012 11:00:00 AM').Caption := 'caption 1';
        Add('6/20/2012 11:00:00 AM','6/20/2012 1:00:00 PM').Caption := 'caption 2';
        Add('6/20/2012 1:00:00 PM','6/20/2012 3:00:00 PM').Caption := 'caption 3';
    end;
end
```

VFP

```
with thisform.Schedule1
.DefaultEventLongLabel = "<%=%5%>"
.DefaultEventShortLabel = .DefaultEventLongLabel
.Calendar.Selection = {^2012-6-20}
with .Events
.Add({^2012-6-20 9:00:00},{^2012-6-20 11:00:00}).Caption = "caption 1"
.Add({^2012-6-20 11:00:00},{^2012-6-20 13:00:00}).Caption = "caption 2"
.Add({^2012-6-20 13:00:00},{^2012-6-20 15:00:00}).Caption = "caption 3"
endwith
endwith
```

```
local oSchedule,var_Event,var_Event1,var_Event2,var_Events
oSchedule = form.Activex1.nativeObject
oSchedule.DefaultEventLongLabel = "<%=%5%>"
oSchedule.DefaultEventShortLabel = oSchedule.DefaultEventLongLabel
oSchedule.Calendar.Selection = "06/20/2012"
var_Events = oSchedule.Events
  // var_Events.Add("06/20/2012 09:00:00","06/20/2012 11:00:00").Caption
= "caption 1"
  var_Event = var_Events.Add("06/20/2012 09:00:00","06/20/2012 11:00:00")
  with (oSchedule)
    TemplateDef = [Dim var_Event]
    TemplateDef = var_Event
    Template = [var_Event.Caption = "caption 1"]
  endwith
  // var_Events.Add("06/20/2012 11:00:00","06/20/2012 13:00:00").Caption
= "caption 2"
  var_Event1 = var_Events.Add("06/20/2012 11:00:00","06/20/2012 13:00:00")
  with (oSchedule)
    TemplateDef = [Dim var_Event1]
    TemplateDef = var_Event1
    Template = [var_Event1.Caption = "caption 2"]
  endwith
  // var_Events.Add("06/20/2012 13:00:00","06/20/2012 15:00:00").Caption
= "caption 3"
  var_Event2 = var_Events.Add("06/20/2012 13:00:00","06/20/2012 15:00:00")
  with (oSchedule)
    TemplateDef = [Dim var_Event2]
    TemplateDef = var_Event2
    Template = [var_Event2.Caption = "caption 3"]
  endwith
```

XBasic (Alpha Five)

Dim oSchedule as P Dim var_Event as P

```
Dim var_Event1 as P
Dim var_Event2 as P
Dim var_Events as P
```

oSchedule = topparent:CONTROL_ACTIVEX1.activex oSchedule.**DefaultEventLongLabel** = "<%=%5%>" oSchedule.**DefaultEventShortLabel** = oSchedule.**DefaultEventLongLabel** oSchedule.Calendar.Selection = {06/20/2012} var_Events = oSchedule.Events

'var_Events.Add({06/20/2012 09:00:00},{06/20/2012 11:00:00}).Caption =
"caption 1"
war_Event = war_Events Add((06/20/2012 09:00:00) (06/20/2012 11:00:00))

```
var_Event = var_Events.Add({06/20/2012 09:00:00},{06/20/2012 11:00:00})
oSchedule.TemplateDef = "Dim var_Event"
oSchedule.TemplateDef = var_Event
oSchedule.Template = "var_Event.Caption = \"caption 1\""
```

'var_Events.Add({06/20/2012 11:00:00},{06/20/2012 13:00:00}).Caption = "caption 2"

var_Event1 = var_Events.Add({06/20/2012 11:00:00},{06/20/2012 13:00:00})
oSchedule.TemplateDef = "Dim var_Event1"
oSchedule.TemplateDef = var_Event1
oSchedule.Template = "var_Event1.Caption = \"caption 2\""

'var_Events.Add({06/20/2012 13:00:00},{06/20/2012 15:00:00}).Caption = "caption 3"

var_Event2 = var_Events.Add({06/20/2012 13:00:00},{06/20/2012 15:00:00})
oSchedule.TemplateDef = "Dim var_Event2"
oSchedule.TemplateDef = var_Event2
oSchedule.Template = "var_Event2.Caption = \"caption 3\""

Visual Objects

local var_Events as IEvents

oDCOCX_Exontrol1:**DefaultEventLongLabel** := "<%=%5%>"

```
oDCOCX_Exontrol1:DefaultEventShortLabel :=
oDCOCX_Exontrol1:DefaultEventLongLabel
oDCOCX_Exontrol1:Calendar:Selection := SToD("20120620")
var_Events := oDCOCX_Exontrol1:Events
var_Events:Add(SToD("20120620 09:00:00"),SToD("20120620 11:00:00")):Caption
:= "caption 1"
var_Events:Add(SToD("20120620 11:00:00"),SToD("20120620 13:00:00")):Caption
:= "caption 2"
var_Events:Add(SToD("20120620 13:00:00"),SToD("20120620 15:00:00")):Caption
:= "caption 3"
```

PowerBuilder

```
OleObject oSchedule,var_Events

oSchedule = ole_1.Object

oSchedule.DefaultEventLongLabel = "<%=%5%>"

oSchedule.DefaultEventShortLabel = oSchedule.DefaultEventLongLabel

oSchedule.Calendar.Selection = 2012-06-20

var_Events = oSchedule.Events

var_Events.Add(DateTime(2012-06-20,09:00:00),DateTime(2012-06-

20,11:00:00)).Caption = "caption 1"

var_Events.Add(DateTime(2012-06-20,11:00:00),DateTime(2012-06-

20,13:00:00)).Caption = "caption 2"

var_Events.Add(DateTime(2012-06-20,13:00:00),DateTime(2012-06-

20,15:00:00)).Caption = "caption 3"
```

method Event.ClearShowStatus ()

Clears the status flag, so the ShowStatusEvent property indicates whether the current event shows or hides its status.

Туре

Description

The ClearShowStatus method clears the ShowStatus flag, if previously the <u>ShowStatus</u> property has been set. By default, the <u>ShowStatusEvent</u> property shows or hides the status part for all events. The <u>ShowStatus</u> property shows or hides the status part of giving event. Use the ClearShowStatus method to allow the <u>ShowStatusEvent</u> property to display the event's status, rather than ShowStatus property.

You can:

- show the status part for all events (<u>ShowStatusEvent</u> on True), and use the ShowStatus (ShowStatus on False) property to hide the status part for specified events only.
- hide the status part for all events (<u>ShowStatusEvent</u> on False), and use the ShowStatus (ShowStatus on True) property to show the status part for specified events only.

method Event.ClearStatusColor ()

Clears the status color flag, so the StatusEventColor property indicates the color to show the event's status.

Туре

Description

The ClearStatusColor method clears the <u>StatusColor</u> property, if previously has been set. By default, the <u>StatusEventColor</u> property indicates the visual appearance to show the status part of all events. The <u>StatusColor</u> property indicates the color to show the status for a giving event. The ClearStatusColor method allows the <u>StatusEventColor</u> property to specify the event's status color, rather than <u>StatusColor</u> property. The <u>ShowStatus</u> property indicates whether the event displays the status part. The <u>ShowEventStatus</u> property indicates whether all events shows the status part.

property Event.Client as Variant

Returns the client area of the event.

Туре	Description
Variant	Returns a safe array array of [x,y,width,height] type, relative to the control's top-left corner of the control

The Client property gets the event's client rectangle as a safe array of [x,y,width,height] type, relative to the control's top-left corner. The Client property returns the [0,0,0,0] safe array If the event is not visible.

property Event.Editable as EditableCaptionEnum

Specifies whether the event's caption is editable.

EditableCaptionEnumAn EditableCaptionEnum expression that specifies the property of the event being edited when event performs an inline editing.	Туре	Description
	EditableCaptionEnum	An EditableCaptionEnum expression that specifies the property of the event being edited when event performs an inline editing.

By default, the Editable property is exEditExtraLabel, which means that the event's <u>ExtraLabel</u> property is edited. The Editable property indicates the property of the event to be edited when user double clicks the event. The <u>AllowEditEvent</u> property specifies the combination of keys that the user can use so the event gets inline editing. The <u>LayoutStartChanging(exScheduleEditEvent)</u> event notifies your application once the inline editing starts. The <u>LayoutEndChanging(exScheduleEditEvent)</u> event notifies your application once the inline editing starts. The control fires the <u>AddEvent</u> event which can be used to initialize the Editable property with a different value.

You can use the <u>EventFromPoint(-1,-1)</u> metod during the <u>LayoutStartChanging(</u> exScheduleEditEvent) to store the event from the cursor to a global member, and when <u>LayoutEndChanging(</u> exScheduleEditEvent) occurs, you can use the previously stored member to identify the event being edited like in the following snippet of code:

Private evEdit As Object

```
Private Sub Schedule1_LayoutStartChanging(ByVal Operation As

EXSCHEDULELibCtl.LayoutChangingEnum)

If (Operation = exScheduleEditEvent) Then

Set evEdit = Schedule1.EventFromPoint(-1, -1)

End If

End Sub

Private Sub Schedule1_LayoutEndChanging(ByVal Operation As

EXSCHEDULELibCtl.LayoutChangingEnum)

If (Operation = exScheduleEditEvent) Then

If Not evEdit Is Nothing Then

Debug.Print "Event: " & evEdit.Handle & " has been edited, and the new caption is:

" & evEdit.ExtraLabel

End If

End If
```

End Sub

The following sample edit and displays the event's LongLabel property only. During the AddEvent event you should call the *Ev.Editable on 3*, in case you need to be applied for any new event created.

VBA (MS Access, Excell...)

```
' AddEvent event - Notifies your application once the a new event is added.
Private Sub Schedule1_AddEvent(ByVal Ev As Object)
End Sub
With Schedule1
  .SelectEventStyle = 48
  .DefaultEventLongLabel = ""
  .DefaultEventShortLabel = ""
  .CreateEventLabel = ""
  .Calendar.Selection = #1/10/2001#
  OnResizeControl = 3073
  With .Events
    With .Add(#1/10/2001 9:00:00 AM#,#1/10/2001 0:30:00 PM#)
      Editable = 3
      .LongLabel = "just your label"
    End With
    With .Add(#1/10/2001 10:00:00 AM#,#1/10/2001 1:00:00 PM#)
      Editable = 3
      .LongLabel = "just another label"
    End With
  End With
End With
```

VB6

```
' AddEvent event - Notifies your application once the a new event is added.
Private Sub Schedule1_AddEvent(ByVal Ev As EXSCHEDULELibCtl.IEvent)
End Sub
```

With Schedule1 .SelectEventStyle = exLinesSolid

```
.DefaultEventLongLabel = ""
  .DefaultEventShortLabel = ""
  .CreateEventLabel = ""
  .Calendar.Selection = #1/10/2001#
  .OnResizeControl = OnResizeControlEnum.exResizePanelRight Or
OnResizeControlEnum.exCalendarFit Or OnResizeControlEnum.exCalendarAutoHide
  With Events
    With .Add(#1/10/2001 9:00:00 AM#,#1/10/2001 0:30:00 PM#)
      .Editable = exEditLongLabel
      .LongLabel = "just your label"
    Fnd With
    With .Add(#1/10/2001 10:00:00 AM#,#1/10/2001 1:00:00 PM#)
      .Editable = exEditLongLabel
      .LongLabel = "just another label"
    End With
  End With
End With
```

VB.NET

' AddEvent event - Notifies your application once the a new event is added. Private Sub Exschedule1_AddEvent(ByVal sender As System.Object,ByVal Ev As exontrol.EXSCHEDULELib.Event) Handles Exschedule1.AddEvent End Sub

With Exschedule1 .SelectEventStyle = exontrol.EXSCHEDULELib.LinesStyleEnum.exLinesSolid .DefaultEventLongLabel = "" .DefaultEventShortLabel = "" .CreateEventLabel = "" .CreateEventLabel = "" .Calendar.Selection = #1/10/2001# .OnResizeControl = exontrol.EXSCHEDULELib.OnResizeControlEnum.exResizePanelRight Or exontrol.EXSCHEDULELib.OnResizeControlEnum.exCalendarFit Or exontrol.EXSCHEDULELib.OnResizeControlEnum.exCalendarFit Or exontrol.EXSCHEDULELib.OnResizeControlEnum.exCalendarAutoHide With .Events With .Add(#1/10/2001 9:00:00 AM#,#1/10/2001 0:30:00 PM#)

```
.Editable = exontrol.EXSCHEDULELib.EditableCaptionEnum.exEditLongLabel
.LongLabel = "just your label"
End With
With .Add(#1/10/2001 10:00:00 AM#,#1/10/2001 1:00:00 PM#)
.Editable = exontrol.EXSCHEDULELib.EditableCaptionEnum.exEditLongLabel
.LongLabel = "just another label"
End With
End With
End With
```

VB.NET for /COM

```
' AddEvent event - Notifies your application once the a new event is added.
Private Sub AxSchedule1_AddEvent(ByVal sender As System.Object, ByVal e As
AxEXSCHEDULELib._IScheduleEvents_AddEventEvent) Handles
AxSchedule1.AddEvent
End Sub
With AxSchedule1
  .SelectEventStyle = EXSCHEDULELib.LinesStyleEnum.exLinesSolid
  .DefaultEventLongLabel = ""
  .DefaultEventShortLabel = ""
  .CreateEventLabel = ""
  .Calendar.Selection = #1/10/2001#
  .OnResizeControl = EXSCHEDULELib.OnResizeControlEnum.exResizePanelRight Or
EXSCHEDULELib.OnResizeControlEnum.exCalendarFit Or
EXSCHEDULELib.OnResizeControlEnum.exCalendarAutoHide
  With .Events
    With .Add(#1/10/2001 9:00:00 AM#,#1/10/2001 0:30:00 PM#)
      .Editable = EXSCHEDULELib.EditableCaptionEnum.exEditLongLabel
      .LongLabel = "just your label"
    End With
    With .Add(#1/10/2001 10:00:00 AM#,#1/10/2001 1:00:00 PM#)
      .Editable = EXSCHEDULELib.EditableCaptionEnum.exEditLongLabel
      .LongLabel = "just another label"
    End With
  End With
```

```
End With
```

C++

```
// AddEvent event - Notifies your application once the a new event is added.
void OnAddEventSchedule1(LPDISPATCH Ev)
{
}
  Copy and paste the following directives to your header file as
  it defines the namespace 'EXSCHEDULELib' for the library: 'ExSchedule 1.0 Control
Library'
  #import <ExSchedule.dll>
  using namespace EXSCHEDULELib;
EXSCHEDULELib::ISchedulePtr spSchedule1 = GetDlgItem(IDC_SCHEDULE1)-
>GetControlUnknown();
spSchedule1->PutSelectEventStyle(EXSCHEDULELib::exLinesSolid);
spSchedule1->PutDefaultEventLongLabel(L"");
spSchedule1->PutDefaultEventShortLabel(L"");
spSchedule1->PutCreateEventLabel(L"");
spSchedule1->GetCalendar()->PutSelection("1/10/2001");
spSchedule1-
> PutOnResizeControl(EXSCHEDULELib::OnResizeControlEnum(EXSCHEDULELib::exResiz
| EXSCHEDULELib::exCalendarFit | EXSCHEDULELib::exCalendarAutoHide));
EXSCHEDULELib::IEventsPtr var_Events = spSchedule1->GetEvents();
  EXSCHEDULELib::IEventPtr var_Event = var_Events->Add("1/10/2001 9:00:00
AM","1/10/2001 12:30:00 PM");
    var_Event->PutEditable(EXSCHEDULELib::exEditLongLabel);
    var_Event->PutLongLabel(L"just your label");
  EXSCHEDULELib::IEventPtr var_Event1 = var_Events->Add("1/10/2001 10:00:00
AM","1/10/2001 1:00:00 PM");
    var_Event1->PutEditable(EXSCHEDULELib::exEditLongLabel);
```

var_Event1->PutLongLabel(L"just another label");

C++ Builder

```
// AddEvent event - Notifies your application once the a new event is added.
void __fastcall TForm1::Schedule1AddEvent(TObject *Sender,Exschedulelib_tlb::IEvent
*Ev)
{
}
Schedule1->SelectEventStyle = Exschedulelib_tlb::LinesStyleEnum::exLinesSolid;
Schedule1->DefaultEventLongLabel = L"";
Schedule1->DefaultEventShortLabel = L"";
Schedule1->CreateEventLabel = L"";
Schedule1->Calendar->set_Selection(TVariant(TDateTime(2001,1,10).operator
double()));
Schedule1->OnResizeControl =
Exschedulelib_tlb::OnResizeControlEnum::exResizePanelRight |
Exschedulelib_tlb::OnResizeControlEnum::exCalendarFit |
Exschedulelib_tlb::OnResizeControlEnum::exCalendarAutoHide;
Exschedulelib_tlb::IEventsPtr var_Events = Schedule1->Events;
  Exschedulelib_tlb::IEventPtr var_Event = var_Events-
>Add(TVariant(TDateTime(2001,1,10,9,00,00,0).operator
double()),TVariant(TDateTime(2001,1,10,12,30,00,0).operator double()));
    var Event->Editable =
Exschedulelib_tlb::EditableCaptionEnum::exEditLongLabel;
    var_Event->LongLabel = L"just your label";
  Exschedulelib_tlb::IEventPtr var_Event1 = var_Events-
>Add(TVariant(TDateTime(2001,1,10,10,00,00,0).operator
double()),TVariant(TDateTime(2001,1,10,13,00,00,0).operator double()));
    var_Event1->Editable =
Exschedulelib_tlb::EditableCaptionEnum::exEditLongLabel;
    var_Event1->LongLabel = L"just another label";
```

C#

// AddEvent event - Notifies your application once the a new event is added.
private void exschedule1_AddEvent(object sender,exontrol.EXSCHEDULELib.Event Ev)

```
}
//this.exschedule1.AddEvent += new
exontrol.EXSCHEDULELib.exg2antt.AddEventEventHandler(this.exschedule1_AddeventEventHandler(this.exschedule1_AddeventEventHandler(this.exschedule1_AddeventEventHandler(this.exschedule1_AddeventEventHandler(this.exschedule1_AddeventEventHandler(this.exschedule1_AddeventEventHandler(this.exschedule1_AddeventEventHandler(this.exschedule1_AddeventEventHandler(this.exschedule1_AddeventEventHandler(this.exschedule1_AddeventEventHandler(this.exschedule1_AddeventEventHandler(this.exschedule1_AddeventEventHandler(this.exschedule1_AddeventEventHandler(this.exschedule1_AddeventEventHandler(this.exschedule1_AddeventEventHandler(this.exschedule1_AddeventEventHandler(this.exschedule1_AddeventEventHandler(this.exschedule1_AddeventEventHandler(this.exschedule1_AddeventEventHandler(this.exschedule1_AddeventEventHandler(this.exschedule1_AddeventEventHandler(this.exschedule1_AddeventEventHandler(this.exschedule1_AddeventEventHandler(this.exschedule1_AddeventEventHandler(this.exschedule1_AddeventEventHandler(this.exschedule1_AddeventEventHandler(this.exschedule1_AddeventEventHandler(this.exschedule1_AddeventEventHandler(this.exschedule1_AddeventEventHandler(this.exschedule1_AddeventEventHandler(this.exschedule1_AddeventEventHandler(this.exschedule1_AddeventEventHandler(this.exschedule1_AddeventEventHandler(this.exschedule1_AddeventEventHandler(this.exschedule1_AddeventEventHandler(this.exschedule1_AddeventEventHandler(this.exschedule1_AddeventEventHandler(this.exschedule1_AddeventEventHandler(this.exschedule1_AddeventEventHandler(this.exschedule1_AddeventEventHandler(this.exschedule1_AddeventEventHandler(this.exschedule1_AddeventEventHandler(this.exschedule1_AddeventEventHandler(this.exschedule1_AddeventEventHandler(this.exschedule1_AddeventEventHandler(this.exschedule1_AddeventEventHandler(this.exschedule1_AddeventEventHandler(this.exschedule1_AddeventEventHandler(this.exschedule1_AddeventEventHandler(this.exschedule1_AddeventEventHandler(this.exschedule1_AddeventEventHandler(this.exschedu
```

```
exschedule1.SelectEventStyle =
exontrol.EXSCHEDULELib.LinesStyleEnum.exLinesSolid;
exschedule1.DefaultEventLongLabel = "";
exschedule1.DefaultEventShortLabel = "";
exschedule1.CreateEventLabel = "";
exschedule1.Calendar.Selection =
Convert.ToDateTime("1/10/2001",System.Globalization.CultureInfo.GetCultureInfo("en-
US"));
exschedule1.OnResizeControl =
exontrol.EXSCHEDULELib.OnResizeControlEnum.exResizePanelRight |
exontrol.EXSCHEDULELib.OnResizeControlEnum.exCalendarFit |
exontrol.EXSCHEDULELib.OnResizeControlEnum.exCalendarAutoHide;
exontrol.EXSCHEDULELib.Events var Events = exschedule1.Events;
  exontrol.EXSCHEDULELib.Event var Event =
var_Events.Add(Convert.ToDateTime("1/10/2001 9:00:00
AM", System. Globalization. CultureInfo. GetCultureInfo("en-
US")),Convert.ToDateTime("1/10/2001 12:30:00
PM",System.Globalization.CultureInfo.GetCultureInfo("en-US")));
    var Event.Editable =
exontrol.EXSCHEDULELib.EditableCaptionEnum.exEditLongLabel;
    var_Event.LongLabel = "just your label";
  exontrol.EXSCHEDULELib.Event var Event1 =
var_Events.Add(Convert.ToDateTime("1/10/2001 10:00:00
AM", System. Globalization. CultureInfo. GetCultureInfo("en-
US")),Convert.ToDateTime("1/10/2001 1:00:00
PM",System.Globalization.CultureInfo.GetCultureInfo("en-US")));
    var Event1.Editable =
exontrol.EXSCHEDULELib.EditableCaptionEnum.exEditLongLabel;
    var_Event1.LongLabel = "just another label";
```

```
<SCRIPT FOR="Schedule1" EVENT="AddEvent(Ev)" LANGUAGE="JScript">
</SCRIPT>
<OBJECT classid = "clsid:9B09E13D-7A88-4299-9DBE-383380435377"
id="Schedule1"></OBJECT>
<SCRIPT LANGUAGE="JScript">
  Schedule1.SelectEventStyle = 48;
  Schedule1.DefaultEventLongLabel = "";
  Schedule1.DefaultEventShortLabel = "";
  Schedule1.CreateEventLabel = "";
  Schedule1.Calendar.Selection = "1/10/2001";
  Schedule1.OnResizeControl = 3073;
 var var_Events = Schedule1.Events;
    var var_Event = var_Events.Add("1/10/2001 9:00:00 AM","1/10/2001 12:30:00
PM");
      var_Event.Editable = 3;
      var_Event.LongLabel = "just your label";
    var var_Event1 = var_Events.Add("1/10/2001 10:00:00 AM","1/10/2001 1:00:00
PM");
      var_Event1.Editable = 3;
      var_Event1.LongLabel = "just another label";
</SCRIPT>
```

C# for /COM

```
// AddEvent event - Notifies your application once the a new event is added.
private void axSchedule1_AddEvent(object sender,
AxEXSCHEDULELib._IScheduleEvents_AddEventEvent e)
{
}
//this.axSchedule1.AddEvent += new
AxEXSCHEDULELib._IScheduleEvents_AddEventEventHandler(this.axSchedule1_f
axSchedule1.SelectEventStyle = EXSCHEDULELib.LinesStyleEnum.exLinesSolid;
```

```
axSchedule1.DefaultEventLongLabel = "";
```

```
axSchedule1.DefaultEventShortLabel = "";
axSchedule1.CreateEventLabel = "";
axSchedule1.Calendar.Selection =
Convert.ToDateTime("1/10/2001",System.Globalization.CultureInfo.GetCultureInfo("en-
US"));
axSchedule1.OnResizeControl =
EXSCHEDULELib.OnResizeControlEnum.exResizePanelRight |
EXSCHEDULELib.OnResizeControlEnum.exCalendarFit |
EXSCHEDULELib.OnResizeControlEnum.exCalendarAutoHide;
EXSCHEDULELib.Events var Events = axSchedule1.Events;
  EXSCHEDULELib.Event var Event =
var Events.Add(Convert.ToDateTime("1/10/2001 9:00:00
AM", System. Globalization. CultureInfo. GetCultureInfo("en-
US")),Convert.ToDateTime("1/10/2001 12:30:00
PM",System.Globalization.CultureInfo.GetCultureInfo("en-US")));
    var_Event.Editable = EXSCHEDULELib.EditableCaptionEnum.exEditLongLabel;
    var_Event.LongLabel = "just your label";
  EXSCHEDULELib.Event var Event1 =
var_Events.Add(Convert.ToDateTime("1/10/2001 10:00:00
AM", System. Globalization. CultureInfo. GetCultureInfo("en-
US")),Convert.ToDateTime("1/10/2001 1:00:00
PM",System.Globalization.CultureInfo.GetCultureInfo("en-US")));
    var_Event1.Editable = EXSCHEDULELib.EditableCaptionEnum.exEditLongLabel;
    var_Event1.LongLabel = "just another label";
```

X++ (Dynamics Ax 2009)

```
// AddEvent event - Notifies your application once the a new event is added.
void onEvent_AddEvent(COM _Ev)
{
    ;
    public void init()
{
        COM com_Event,com_Event1,com_Events;
    }
}
```

```
anytype var_Event,var_Event1,var_Events;
  super();
  exschedule1.SelectEventStyle(48/*exLinesSolid*/);
  exschedule1.DefaultEventLongLabel("");
  exschedule1.DefaultEventShortLabel("");
  exschedule1.CreateEventLabel("");
exschedule1.Calendar().Selection(COMVariant::createFromDate(str2Date("1/10/2001",2
  exschedule1.OnResizeControl(3073/*exResizePanelRight | exCalendarFit |
exCalendarAutoHide*/);
  var_Events = exschedule1.Events(); com_Events = var_Events;
    var Event =
com_Events.Add(COMVariant::createFromUtcDateTime(str2Datetime("1/10/2001
9:00:00",213)),COMVariant::createFromUtcDateTime(str2Datetime("1/10/2001
12:30:00",213))); com_Event = var_Event;
      com_Event.Editable(3/*exEditLongLabel*/);
      com_Event.LongLabel("just your label");
    var Event1 =
com_Events.Add(COMVariant::createFromUtcDateTime(str2Datetime("1/10/2001
10:00:00",213)),COMVariant::createFromUtcDateTime(str2Datetime("1/10/2001
13:00:00",213))); com_Event1 = var_Event1;
      com_Event1.Editable(3/*exEditLongLabel*/);
      com_Event1.LongLabel("just another label");
```

Delphi 8 (.NET only)

```
// AddEvent event - Notifies your application once the a new event is added.
procedure TWinForm1.AxSchedule1_AddEvent(sender: System.Object; e:
AxEXSCHEDULELib._IScheduleEvents_AddEventEvent);
begin
end;
```

```
with AxSchedule1 do
begin
  SelectEventStyle := EXSCHEDULELib.LinesStyleEnum.exLinesSolid;
  DefaultEventLongLabel := ";
  DefaultEventShortLabel := ";
  CreateEventLabel := ";
  Calendar.Selection := \frac{1}{10}/2001;
  OnResizeControl :=
Integer(EXSCHEDULELib.OnResizeControlEnum.exResizePanelRight) Or
Integer(EXSCHEDULELib.OnResizeControlEnum.exCalendarFit) Or
Integer(EXSCHEDULELib.OnResizeControlEnum.exCalendarAutoHide);
  with Events do
  begin
    with Add('1/10/2001 9:00:00 AM','1/10/2001 12:30:00 PM') do
    begin
      Editable := EXSCHEDULELib.EditableCaptionEnum.exEditLongLabel;
      LongLabel := 'just your label';
    end;
    with Add('1/10/2001 10:00:00 AM','1/10/2001 1:00:00 PM') do
    begin
      Editable := EXSCHEDULELib.EditableCaptionEnum.exEditLongLabel;
      LongLabel := 'just another label';
    end;
  end;
end
```

Delphi (standard)

```
// AddEvent event - Notifies your application once the a new event is added.
procedure TForm1.Schedule1AddEvent(ASender: TObject; Ev : IEvent);
begin
end;
with Schedule1 do
begin
SelectEventStyle := EXSCHEDULELib_TLB.exLinesSolid;
DefaultEventLongLabel := ";
```

```
DefaultEventShortLabel := ";
  CreateEventLabel := ";
  Calendar.Selection := '1/10/2001';
  OnResizeControl := Integer(EXSCHEDULELib_TLB.exResizePanelRight) Or
Integer(EXSCHEDULELib_TLB.exCalendarFit) Or
Integer(EXSCHEDULELib_TLB.exCalendarAutoHide);
  with Events do
  begin
    with Add('1/10/2001 9:00:00 AM','1/10/2001 12:30:00 PM') do
    begin
      Editable := EXSCHEDULELib_TLB.exEditLongLabel;
      LongLabel := 'just your label';
    end;
    with Add('1/10/2001 10:00:00 AM','1/10/2001 1:00:00 PM') do
    begin
      Editable := EXSCHEDULELib_TLB.exEditLongLabel;
      LongLabel := 'just another label';
    end;
  end;
end
```

VFP

```
*** AddEvent event - Notifies your application once the a new event is added. ***
LPARAMETERS Ev
```

```
with thisform.Schedule1
.SelectEventStyle = 48
.DefaultEventLongLabel = ""
.DefaultEventShortLabel = ""
.CreateEventLabel = ""
.Calendar.Selection = {^2001-1-10}
.OnResizeControl = 3073
with .Events
with .Add({^2001-1-10 9:00:00},{^2001-1-10 12:30:00})
.Editable = 3
.LongLabel = "just your label"
```

```
endwith
with .Add({^2001-1-10 10:00:00},{^2001-1-10 13:00:00})
.Editable = 3
.LongLabel = "just another label"
endwith
endwith
```

dBASE Plus

```
/*
with (this.ACTIVEX1.nativeObject)
  AddEvent = class::nativeObject_AddEvent
endwith
*/
// Notifies your application once the a new event is added.
function nativeObject_AddEvent(Ev)
  local oSchedule
  oSchedule = form.Activex1.nativeObject
return
local oSchedule,var_Event,var_Event1,var_Events
oSchedule = form.Activex1.nativeObject
oSchedule.SelectEventStyle = 48
oSchedule.DefaultEventLongLabel = ""
oSchedule.DefaultEventShortLabel = ""
oSchedule.CreateEventLabel = ""
oSchedule.Calendar.Selection = "01/10/2001"
oSchedule.OnResizeControl = 3073 /*exResizePanelRight | exCalendarFit |
exCalendarAutoHide*/
var Events = oSchedule.Events
  var_Event = var_Events.Add("01/10/2001 09:00:00","01/10/2001 12:30:00")
    var Event.Editable = 3
    var_Event.LongLabel = "just your label"
  var_Event1 = var_Events.Add("01/10/2001 10:00:00","01/10/2001 13:00:00")
    var Event1.Editable = 3
```

```
var_Event1.LongLabel = "just another label"
```

XBasic (Alpha Five)

```
' Notifies your application once the a new event is added.
function AddEvent as v (Ev as OLE::Exontrol.Schedule.1::IEvent)
  Dim oSchedule as P
  oSchedule = topparent:CONTROL_ACTIVEX1.activex
end function
Dim oSchedule as P
Dim var_Event as P
Dim var_Event1 as P
Dim var Events as P
oSchedule = topparent:CONTROL_ACTIVEX1.activex
oSchedule.SelectEventStyle = 48
oSchedule.DefaultEventLongLabel = ""
oSchedule.DefaultEventShortLabel = ""
oSchedule.CreateEventLabel = ""
oSchedule.Calendar.Selection = {01/10/2001}
oSchedule.OnResizeControl = 3073 'exResizePanelRight + exCalendarFit +
exCalendarAutoHide
var Events = oSchedule.Events
  var_Event = var_Events.Add({01/10/2001 09:00:00},{01/10/2001 12:30:00})
    var Event.Editable = 3
    var_Event.LongLabel = "just your label"
  var_Event1 = var_Events.Add({01/10/2001 10:00:00},{01/10/2001 13:00:00})
    var Event1.Editable = 3
    var_Event1.LongLabel = "just another label"
```

Visual Objects

METHOD OCX_Exontrol1AddEvent(Ev) CLASS MainDialog

// AddEvent event - Notifies your application once the a new event is added.

```
RETURN NIL
```

```
local var_Event,var_Event1 as IEvent
local var_Events as IEvents
oDCOCX_Exontrol1:SelectEventStyle := exLinesSolid
oDCOCX_Exontrol1:DefaultEventLongLabel := ""
oDCOCX_Exontrol1:DefaultEventShortLabel := ""
oDCOCX_Exontrol1:CreateEventLabel := ""
oDCOCX_Exontrol1:Calendar:Selection := SToD("20010110")
oDCOCX_Exontrol1:OnResizeControl := exResizePanelRight | exCalendarFit |
exCalendarAutoHide
var Events := oDCOCX Exontrol1:Events
  var_Event := var_Events:Add(SToD("20010110 09:00:00"),SToD("20010110
12:30:00"))
    var_Event:Editable := exEditLongLabel
    var_Event:LongLabel := "just your label"
  var_Event1 := var_Events:Add(SToD("20010110 10:00:00"),SToD("20010110
13:00:00"))
    var_Event1:Editable := exEditLongLabel
    var_Event1:LongLabel := "just another label"
```

PowerBuilder

```
/*begin event AddEvent(oleobject Ev) - Notifies your application once the a new
event is added.*/
/*
    OleObject oSchedule
    oSchedule = ole_1.Object
*/
/*end event AddEvent*/
OleObject oSchedule,var_Event,var_Event1,var_Events
    C = b = b = 1.00 is is
```

```
oSchedule = ole_1.Object
oSchedule.SelectEventStyle = 48
```

```
oSchedule.DefaultEventLongLabel = ""
oSchedule.DefaultEventShortLabel = ""
oSchedule.CreateEventLabel = ""
oSchedule.Calendar.Selection = 2001-01-10
oSchedule.OnResizeControl = 3073 /*exResizePanelRight | exCalendarFit |
exCalendarAutoHide*/
var_Events = oSchedule.Events
var_Event = var_Events.Add(DateTime(2001-01-10,09:00:00),DateTime(2001-01-
10,12:30:00))
var_Event.Editable = 3
var_Event.LongLabel = "just your label"
var_Event1 = var_Events.Add(DateTime(2001-01-10,10:00:00),DateTime(2001-01-
10,13:00:00))
var_Event1.Editable = 3
```

var_Event1.LongLabel = "just another label"

property Event.End as Date

Specifies the ending date/time of the event.

Туре	Description
Date	A DATE expression that specifies the upper margin of the event

The End property of the Event indicates the date/time when the event or the appointment ends. The End parameter of the <u>Add</u> method indicates the ending point of the event at adding time. The <u>Start</u> property of the event indicates the starting point of the event. The Start and End properties may be identical if the <u>AllDayEvent</u> property is True. The <u>UpdateEvent</u> event is fired once the End property is changed. The <u>Resizable</u> property of the Event indicates whether the user can resize the event at runtime (start, end or both). The <u>Movable</u> property of the Event indicates whether the user can move the event at runtime. You can use the <u>MinDate/MaxDate</u> property specifies the range of dates where the <u>Start/End</u> can be shown. You can use the <u>MoveBy</u> method to delay the current event with a specified value time. You can use the <u>KnownProperty</u>(exEventDuration) to change the event's duration.

The <u>KnownProperty</u>(exEventEndDateTime) property indicates the End property on a label property such as: <u>DefaultEventLongLabel</u>, <u>DefaultEventShortLabel</u>, <u>CreateEventLabel</u>, <u>UpdateEventsLabel</u>, <u>ShortLabel</u>, <u>LongLabel</u> and <u>ExtraLabel</u>.

- You can use the <u>KnownProperty</u>(exEventStartDate)/<u>KnownProperty</u>(exEventStartTime) property to extract the starting date/time of the event.
- You can use the <u>KnownProperty</u>(exEventEndDate)/<u>KnownProperty</u>(exEventEndTime) property to extract the ending date/time of the event.
- You can use the <u>KnownProperty</u>(exEventDuration) property to specifies the duration/length of the event.

method Event.EndUpdateEvent (StartUpdateEvent as Long)

Adds programmatically updated properties of the calendar-event to undo/redo queue.

Туре

Description

StartUpdateEvent as Long A long expression that specifies the handle being returned by the <u>StartUpdateEvent</u> property

The <u>StartUpdateEvent</u>/EndUpdateEvent methods record and add changes of the current calendar-event to the control's Undo/Redo queue. You can use the <u>StartBlockUndoRedo</u> / <u>EndBlockUndoRedo</u> methods to group multiple Undo/Redo operations into a single-block. The <u>AllowUndoRedo</u> property specifies whether the control supports undo/redo operations for objects (calendar-events). No entry is added to the Undo/Redo queue if no property is changed for the current calendar-event. Each call of the <u>StartUpdateEvent</u> must be succeeded by a EndUpdateEvent call. The <u>UndoListAction</u> property lists the Undo actions that can be performed in the chart. The <u>RedoListAction</u> property lists the Redo actions that can be performed in the chart.

The <u>StartUpdateEvent</u>/EndUpdateEvent methods can record changes for all properties from 1 (exEventStartDateTime) to EXEVENTMAX listed by <u>EventKnownPropertyEnum</u> type.

The Undo/Redo records show as:

 "UpdateEvent; EVENTID", indicates that one or more properties of the calendar-event has been updated, using the <u>StartUpdateEvent</u> / EndUpdateEvent methods

within the UndoListAction/RedoListAction result

method Event.EnsureVisible ()

Scrolls the control to ensure that the current calendar-event fits the control's visible area.

Туре

Description

The EnsureVisible method scrolls the control to ensure that the current calendar-event fits the control's visible area.

property Event.ExtraLabel as String

Specifies the extra label to be displayed on the event.

Туре	Description
String	A string expression that specifies the extended HTML label, to be displayed on the event's body.

By default, the ExtraLabel property is empty, which indicates no extra label is being displayed. The event displays the ExtraLabel, only if the LongLabel property is displayed. The LongLabel property is displayed only if it fits the event's body, else the ShortLabel property is shown. For instance, the ShortLabel property is shown if the event's body is too small. The AddEvent event occurs once a new event is added to the Events collection. You can handle the AddEvent event to initialize the ExtraLabel with a different value. The Editable property of the event indicates the property of the event being inline edited. BY default, the event's ExtraLabel property is inline edited when the user double clicks the event. The ExtraLabelAlign property specifies the alignment of the event, relative to event's borders. The KnowProperty(exEventExtraLabel) is equivalent with the ExtraLabel property.

Here's a few samples:

- "new", simple new text is shown.
- "<a no>title", displays a clickable text such as <u>title</u>, and <u>AnchorClick</u> can be used to determine whether the *no* anchor has been clicked.
- "<a>pic1:32", displays a click able image, the <u>AnchorClick</u> can be used to determine whether the anchor has been clicked. We would recommend using the <u>Pictures</u> or <u>ExtraPictures</u> property to assign pictures to an event.
- "<%=%256%>", displays the event's start and end points in a short format.
- "<%=%257%>", displays the event's margins in a long format.
- "Start: <%=%1%>
End: <%=%2%>", displays the starting margin of the even on the first line, while on the second line it displays the ending point of the event.
- "<%=%256%>
Caption: <%=%5%>", displays the event's margins in short format on the first line, and on the second line it displays the event's <u>Caption</u> property. The caption shown on the event's body is automatically updated once the event is moved to a new position or the event's Caption is changed.
- "<%=%256%>
<%=%264? `repetitive event`:``%>" displays automatically the "repetitive event" for repetitive events, or when the event's <u>Repetitive</u> property is not empty and valid
- "Duration: <%=((1:=int(0:= (date(%2)-date(%1)))) != 0 ? (=:1 + ' day(s)') : ") + (=:1 ? ' ' : ") + ((1:=int(0:=((=:0 =:1 + 1/24/60/60/2)*24))) != 0 ? =:1 + ' hour(s)' : ") + (=:1 ? ' ' : ") + ((1:=round((=:0 =:1)*60)) != 0 ? =:1 + ' min(s)' : ")%>" displays the duration

of the event in days, hours and minutes.

- "<%=%256%>
Duration: <%=((1:=int(0:= (date(%2)-date(%1)))) != 0 ? (=:1 + ' day(s)') : ") + (=:1 ? ' ' : ") + ((1:=int(0:=((=:0 =:1 + 1/24/60/60/2)*24))) != 0 ? =:1 + ' hour(s)' : ") + (=:1 ? ' ' : ") + ((1:=round((=:0 =:1)*60)) != 0 ? =:1 + ' min(s)' : ")%>" displays the event's margins on the first line and the duration of the event in days, hours and minutes, on the second line
- "<%=%><%=%5%>
<%=%256%>", displays the event's Caption on first line(s), following by the event's Start/End margins in short date-time format. The <%=%> prefix forces the expression to be re-evaluated and apply any HTML tag found. For instance, %5 indicates the event's Caption property, and if it contains HTML tags they will be applied as is, instead displaying them as a plain text. Any expression that starts with "<%=%>" is re- evaluated and its result is displayed in HTML format (available starting with the version 12.2)

The <u>EventKnowPropertyEnum</u> defines the %identifiers that can be used in formula <%=FORMULA%>. For instance, the ExtraLabel property on "Start: <%=time(%1) replace `AM` with ``%>" displays the time when the event starts with no AM time indicators.

The property supports the following identifiers. These identifiers can be used in FORMULA format:

- **%1**, Indicates the starting date/time of the event as DATE type, equivalent with <u>Start</u> property
- %2, Indicates the ending date/time of the event as DATE type, equivalent with End property
- %3, Indicates if the current event is an all day event as BOOL type, equivalent with <u>AllDayEvent</u> property
- %4, Indicates the identifier of the event's group, as LONG type, equivalent with <u>GroupID</u> property.
- **%5**, Indicates the caption of the event, as STRING expression, equivalent with <u>Caption</u> property.
- %6, Indicates the extra data associated with the event, as VARIANT type, equivalent with <u>UserData</u> property.
- %7, Gets or sets the duration of the event as FLOAT expression. Above you can find how you can display the duration of the event in hours, minutes...
- %8, Specifies the repetitive expression of the event, equivalent with <u>Repetitive</u> property.
- %256, Gets the margins of the event in a short format, as a STRING expression. The <u>ShortDateFormat</u> property defines the short date format. The <u>ShortTimeFormat</u> property defines the short time format.
- %257, Gets the margins of the event in a long format, as a STRING expression. The LongDateFormat property defines the long date format. The LongTimeFormat property

defines the long time format.

- %258, Gets the starting date (not including the time) of the current event, as a DATE type.
- %259, Gets the starting time (not including the date) of the current event, as DATE type from 0 to 1.
- %260, Gets the ending date (not including the time) of the current event, as a DATE type.
- **%261**, Gets the ending time (not including the date) of the current event, as DATE type from 0 to 1.
- %262, Gets the label of the owner group, as STRING expression, equivalent with <u>Caption</u> property of the Group's event.
- %263, Gets the title of the owner group, as STRING expression, equivalent with <u>Title</u> property of the Group's event.
- %264, Indicates if the current event is a repetitive event, as BOOL type. This flag returns TRUE, if the <u>Repetitive</u> property is not empty, and the expression is valid.

The FORMULA, is identified by <%=FORMULA%>, and supports the following predefined operators and functions:

The supported binary arithmetic operators are:

- * (multiplicity operator), priority 5
- *I* (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:

- $\circ~$ 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
- Itrim (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 property supports the following built-in HTML tags:

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

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

- exp, stores the plain text to be shown once the user clicks the anchor, such as "<a ;exp=show lines>"
- e64, encodes in BASE64 the HTML text to be shown once the user clicks the anchor, such as "<a

;e64=gA8ABmABnABjABvABshIAOQAEAAHAAGESikWio+ABzABohp3iELABpABu " that displays show lines- in gray when the user clicks the + anchor. The "gA8ABmABnABjABvABshIAOQAEAAHAAGESikWio+ABzABohp3iELABpABuABljY string encodes the "<fgcolor 808080>show lines<a>-</fgcolor>" The Decode64Text/Encode64Text methods of the eXPrint can be used to decode/encode e64 fields.

Any ex-HTML caption can be transformed to an expandable-caption, by inserting the anchor ex-HTML tag. For instance, "<solidline>Header</solidline>

Line1<r><a ;exp=show lines>+
Line2
Line3" shows the Header in underlined and bold on the first line and Line1, Line2, Line3 on the rest. The "show lines" is shown instead of Line1, Line2, Line3 once the user clicks the + sign.

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

- **<fgcolor rrggbb>** ... **</fgcolor>** or **<**fgcolor=rrggbb> ... **</**fgcolor> displays text with a specified foreground color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- <bgcolor rrggbb> ... </bgcolor> or <bgcolor=rrggbb> ... </bgcolor> displays text with a specified background color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- <solidline rrggbb> ... </solidline> or <solidline=rrggbb> ... </solidline> draws a solidline 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
- number[:width] 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.
- key[:width] 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 & (&), &It; (<), > (>), &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 FFFFF;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" generates the following picture:

shadow

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

outline anti-aliasing

property Event.ExtraLabelAlign as ContentAlignmentEnum

Indicates the alignment of the event's extra label.

Туре	Description
<u>ContentAlignmentEnum</u>	A ContentAlignmentEnum expression that specifies the alignment of the ExtraLabel property of the Event. The ExtraLabelAlign property supports additionally the following flag:
	 exWidth (4), to distribute the text on the element's width

By default, the ExtraLabelAlign property is exBottomLeft. The ExtraLabelAlign property aligns the event's <u>ExtraLabel</u> property on the event's body. The event displays the ExtraLabel, only if the <u>LongLabel</u> property is displayed. The <u>LongLabel</u> property is displayed only if it fits the event's body, else the <u>ShortLabel</u> property is shown. For instance, the ShortLabel property is shown if the event's body is too small. The <u>AddEvent</u> event occurs once a new event is added to the Events collection. You can handle the AddEvent event to initialize the ExtraLabel/ExtraLabelAlign with a different value.

property Event.ExtraPictures as String

Specifies the list of extra pictures to be displayed on the event.

Туре	Description
String	A string expression that specifies the list of pictures to be shown on the event's body. The event's body can display one ore more pictures at the time, on different lines. For instance: "1,2/pic1" displays the 1 and 2 icons on the first line, while pic2 is displayed on the second line.

By default, the ExtraPictures property is "", which means that initially no pictures are being displayed on the event. The <u>Pictures</u> or/and ExtraPictures property displays a collection of icons, pictures in the event's body. The pictures are shown on the event's body only if they fit the event's body. For instance, if the event is too small, the <u>ShortLabel</u> is displayed, and no icons or pictures are displayed. The <u>ExtraPicturesAlign</u> property indicates the alignment of the extra pictures relative to the event's borders.

The <u>Images</u> method loads icons to the control, <u>HTMLPicture</u> assigns a key to a picture object, and the <u>Pictures</u> collection handles the identifiers of the pictures that can be used in the <u>Pictures</u> or ExtraPictures properties. The <u>AddEvent</u> event notifies your application once a new event is added. You can use this event to initialize the Pictures/PicturesAlign or ExtraPicturesAlign properties.

In conclusion, in order to display an icon or a picture in the control you need first to load the icons or the pictures you plan to display, using the <u>Images</u> method, <u>HTMLPicture</u>, or <u>Add</u> method of the ExPictures collection. The Images collection can display only 16x16 icons, while the <u>HTMLPicture</u>, or <u>Add</u> method can load and display custom sized pictures. The <u>Width/Height</u> property specifies the width and height of the picture to be displayed in the event's body.

The event can display icons, pictures several ways as follows:

- Using the HTML tag, if used in any label properties such as LongLabel or ExtraLabel property. The ShortLabel property can not display images or HTML font attributes. If the tag is included in a <a> HTML tag, you have a clickable image through the <u>AnchorClick</u> event. This option can be used to display a default icon or picture for all events in the control using the <u>DefaultEventLongLabel</u> property.
- Using the <u>Pictures</u> property of the Event object, which indicates a list of pictures from Pictures collection to be displayed on the event's body. The <u>PicturesAlign</u> property indicates the alignment of the pictures relative to the borders of the event. The hand cursor is shown, if the ExPicture.<u>ShowCursorHand</u> property is set, when the cursor hovers the picture. The <u>PictureClick</u> event is fired if the user clicks a picture on event's picture. You can use the <u>PictureFromPoint</u> property to get the identifier of the picture

from the cursor.

• Using the **ExtraPictures** property of the Event object, which indicates a list of pictures from Pictures collection to be displayed on the event's body. The <u>ExtraPicturesAlign</u> property indicates the alignment of the pictures relative to the borders of the event. The hand cursor is shown, if the ExPicture.<u>ShowCursorHand</u> property is set, when the cursor hovers the picture. The <u>PictureClick</u> event is fired if the user clicks a picture on event's picture. You can use the <u>PictureFromPoint</u> property to get the identifier of the picture from the cursor.

The Picture and ExtraPictures may display one or more pictures at the time. The, character indicates the separator of pictures in the same line, while the *I* character divides the lines to show the pictures. For instance, "1,2" displays icon with the index 1 and 2 on the same line, while the "1/2,pic1" displays the first icon on the first line, the second icon and the picture pic1 on the second line.

property Event.ExtraPicturesAlign as ContentAlignmentEnum

Indicates the alignment of the event's extra picture.

Туре	Description
<u>ContentAlignmentEnum</u>	A ContentAlignmentEnum expression that specifies the alignment of the pictures being displayed by the ExtraPictures property.

By default, the ExtraPicturesAlign property is exTopRight, which means any extra picture associated to the event is displayed on the top-right corner of the event. The ExtraPicturesAlign property has effect only if the ExtraPictures property refers valid pictures, and the event's body has enough space to display the pictures. The AddEvent event notifies your application once a new event is added. You can use this event to initialize the Pictures/PicturesAlign or ExtraPictures/ExtraPicturesAlign properties. The Width/Height property specifies the width and height of the picture to be displayed in the event's body.

property Event.GroupID as Long

Specifies the identifier of the group where the Event object belongs.

Туре	Description
Long	A Long expression that specifies the identifier of the group where the event belongs.

By default the GroupID property is 1. The GroupID property has effect when you need to display events or appointments on different groups. The <u>Item</u> property of the Groups collection can be used to access the <u>Group</u> object based on its identifier. The associated Group of the event may handle the colors, patter to display the event using the <u>EventBackColor</u>, <u>EventForeColor</u>, <u>EventPattern</u> properties. The group colors are being applied if the <u>ApplyGroupingColors</u> property is True. When the user moves an event from a group to another, at runtime, the GroupID property may be changed, and the <u>UpdateEvent</u> event occurs.

The control displays groups if:

- <u>ShowGroupingEvents</u> property is True
- The <u>Groups</u> collection has elements. By default, the Groups collection contains no Group objects. Use the <u>Add</u> method of the Groups collection to add new groups to the control.
- The Groups collection contains visible elements (Visible on True)

If the control displays groups the GroupID property of the newly created event is automatically updated with the group where the event has been created. The <u>AllowMoveEventToOtherGroup</u> property specifies whether the user can move an event from a group to another at runtime. The <u>AllowMoveEventToOtherGroup</u> property on False, prevents moving events from a group to another, at runtime.

property Event.Handle as Long

Gets handle of the Event object.

Туре	Description
Long	A Long expression that specifies the handle of the event.

The Handle property indicates the handle of the event, and it is unique while the event is living. The Handle value is automatically allocated by the control as soon as the event is being created. The <u>AddEvent</u> event occurs once a new event is created/added. The <u>Start/End</u> properties of the event specifies the margins of the event. The Handle property is read only, so the user can not change it. You can use the <u>UserData</u> property to associate any extra data to the event. The <u>KnownProperty</u>(exEventID) property specifies the event's identifier. The Handle is automatically generated by the control, and can not be changed, while the IDentifier can be set by the user.

property Event.KnownProperty(Property as EventKnownPropertyEnum) as Variant

Specifies the value for the Event's property giving its identifier.

Туре	Description
Property as <u>EventKnownPropertyEnum</u>	A Property being requested
Variant	A VARIANT expression that specifies the value of the requested property.

The KnownProperty property may access a property of the event giving its identifier. For instance, the KnownProperty(exEventRepetitive) property indicates whether the current event is repetitive or not. The property returns True, if the <u>Repetitive</u> property is not empty and valid.

The <u>EventKnowPropertyEnum</u> defines the %identifiers that can be used in formula <%=FORMULA%>, in any of the following label properties

- <u>DefaultEventLongLabel</u>, defines the HTML labels for events, when it fit entirely in the event's body.
- <u>DefaultEventShortLabel</u>, defines the labels for events (no HTML attribute is applied), when it does not fit the event's body
- CreateEventLabel, defines the label when creating a new event by dragging
- UpdateEventsLabel, defines the label of the events being moved or resized at runtime
- Event.<u>ShortLabel</u>, defines the event's short label, or the label to be shown when the LongLabel does not fit entirely the event's body. The ShortLabel displays no HTML tags
- Event.<u>LongLabel</u>, defines the event's HTML long label, when it fits the body. If the LongLabel does not fit entirely the event's body, the ShortLabel is displayed instead.
- Event.<u>ExtraLabel</u>, defines the event's extra HTML label. The event's ExtraLabel is displayed ONLY, if the LongLabel fits the event's body

The label properties supports the following identifiers. These identifiers can be used in FORMULA expression:

- **%1**, Indicates the starting date/time of the event as DATE type, equivalent with <u>Start</u> property
- %2, Indicates the ending date/time of the event as DATE type, equivalent with <u>End</u> property
- %3, Indicates if the current event is an all day event as BOOL type, equivalent with <u>AllDayEvent</u> property
- %4, Indicates the identifier of the event's group, as LONG type, equivalent with <u>GroupID</u> property.

- **%5**, Indicates the caption of the event, as STRING expression, equivalent with <u>Caption</u> property.
- %6, Indicates the extra data associated with the event, as VARIANT type, equivalent with <u>UserData</u> property.
- %7, Gets or sets the duration of the event as FLOAT expression. Above you can find how you can display the duration of the event in hours, minutes...
- %8, Specifies the repetitive expression of the event, equivalent with <u>Repetitive</u> property.
- %256, Gets the margins of the event in a short format, as a STRING expression. The <u>ShortDateFormat</u> property defines the short date format. The <u>ShortTimeFormat</u> property defines the short time format.
- %257, Gets the margins of the event in a long format, as a STRING expression. The <u>LongDateFormat</u> property defines the long date format. The <u>LongTimeFormat</u> property defines the long time format.
- %258, Gets the starting date (not including the time) of the current event, as a DATE type.
- %259, Gets the starting time (not including the date) of the current event, as DATE type from 0 to 1.
- **%260**, Gets the ending date (not including the time) of the current event, as a DATE type.
- %261, Gets the ending time (not including the date) of the current event, as DATE type from 0 to 1.
- %262, Gets the label of the owner group, as STRING expression, equivalent with <u>Caption</u> property of the Group's event.
- %263, Gets the title of the owner group, as STRING expression, equivalent with <u>Title</u> property of the Group's event.
- %264, Indicates if the current event is a repetitive event, as BOOL type. This flag returns TRUE, if the <u>Repetitive</u> property is not empty, and the expression is valid.

For instance, the LongLabel = "Start: <%=%1%>
End: <%=%2%>", displays the starting margin of the event on the first line, while on the second line it displays the ending point of the event.

property Event.LabelAlign as ContentAlignmentEnum

Indicates the alignment of the event's long label.

Туре	Description
<u>ContentAlignmentEnum</u>	An ContentAlignmentEnum expression that specifies the ShortLabel/LongLabel alignment. The LabelAlign property supports additionally the following flag:
	 exWidth (4), to distribute the text on the element's width

By default, the LabelAlign property is exTopLeft. The LabelAlign property aligns the event's <u>ShortLabel/LongLabel</u> property on the event's body. The <u>ShortLabel</u> property is shown if the event's body is too small. The event displays the ExtraLabel, only if the <u>LongLabel</u> property is displayed. The <u>LongLabel</u> property is displayed only if it fits the event's body, else the <u>ShortLabel</u> property is shown. The <u>AddEvent</u> event occurs once a new event is added to the Events collection. You can handle the AddEvent event to initialize the LabelAlign with a different value.

The <u>CreateEventLabelAlign</u> property aligns the label being shown when the user creates a new event. The <u>UpdateEventsLabelAlign</u> property aligns the label being shown when the user moves or resizes the events.

property Event.LongLabel as String

Specifies the long label to be displayed on the event.

Туре	Description
String	A string expression that specifies the extended HTML label, to be displayed on the event's body.

By default, the LongLabel property is initialized with the value of the <u>DefaultEventLongLabel</u> property. The LongLabel property is displayed only if it fits the event's body, else the <u>ShortLabel</u> property is shown. For instance, the ShortLabel property is shown if the event's body is too small. The event displays the <u>ExtraLabel</u>, only if the LongLabel property is displayed. The <u>AddEvent</u> event occurs once a new event is added to the Events collection. The <u>LabelAlign</u> property specifies the alignment of the long label. The <u>DefaultEventPadding</u> property indicates the padding of the labels on the event, relative to event's borders. The <u>KnowProperty</u>(exEventLongLabel) is equivalent with the LongLabel property.

Here's a few samples:

- "new", simple new text is shown.
- "<a no>title", displays a clickable text such as <u>title</u>, and <u>AnchorClick</u> can be used to determine whether the *no* anchor has been clicked.
- "<a>pic1:32", displays a click able image, the <u>AnchorClick</u> can be used to determine whether the anchor has been clicked. We would recommend using the <u>Pictures</u> or <u>ExtraPictures</u> property to assign pictures to an event.
- "<%=%256%>", displays the event's start and end points in a short format.
- "<%=%257%>", displays the event's margins in a long format.
- "Start: <%=%1%>
End: <%=%2%>", displays the starting margin of the even on the first line, while on the second line it displays the ending point of the event.
- "<%=%256%>
Caption: <%=%5%>", displays the event's margins in short format on the first line, and on the second line it displays the event's <u>Caption</u> property. The caption shown on the event's body is automatically updated once the event is moved to a new position or the event's Caption is changed.
- "<%=%256%>
<%=%264? `repetitive event`:``%>" displays automatically the "repetitive event" for repetitive events, or when the event's <u>Repetitive</u> property is not empty and valid
- "Duration: <%=((1:=int(0:= (date(%2)-date(%1)))) != 0 ? (=:1 + ' day(s)') : ") + (=:1 ? ' ' : ") + ((1:=int(0:=((=:0 =:1 + 1/24/60/60/2)*24))) != 0 ? =:1 + ' hour(s)' : ") + (=:1 ? ' : ") + ((1:=round((=:0 =:1)*60)) != 0 ? =:1 + ' min(s)' : ")%>" displays the duration of the event in days, hours and minutes.
- "<%=%256%>
Duration: <%=((1:=int(0:= (date(%2)-date(%1)))) != 0 ? (=:1 + '
 day(s)') : ") + (=:1 ? ' ' : ") + ((1:=int(0:=((=:0 =:1 + 1/24/60/60/2)*24))) != 0 ? =:1 +
 ' hour(s)' : ") + (=:1 ? ' ' : ") + ((1:=round((=:0 =:1)*60)) != 0 ? =:1 + ' min(s)' :</pre>

")%>" displays the event's margins on the first line and the duration of the event in days, hours and minutes, on the second line

"<%=%><%=%5%>
<%=%256%>", displays the event's Caption on first line(s), following by the event's Start/End margins in short date-time format. The <%=%> prefix forces the expression to be re-evaluated and apply any HTML tag found. For instance, %5 indicates the event's Caption property, and if it contains HTML tags they will be applied as is, instead displaying them as a plain text. Any expression that starts with "<%=%>" is re- evaluated and its result is displayed in HTML format (available starting with the version 12.2)

The <u>EventKnowPropertyEnum</u> defines the %identifiers that can be used in formula <%=FORMULA%>. For instance, the Label property on "Start: <%=time(%1) replace `AM` with ``%>" displays the time when the event starts with no AM time indicators.

The property supports the following identifiers. These identifiers can be used in FORMULA format:

- **%1**, Indicates the starting date/time of the event as DATE type, equivalent with <u>Start</u> property
- %2, Indicates the ending date/time of the event as DATE type, equivalent with <u>End</u> property
- %3, Indicates if the current event is an all day event as BOOL type, equivalent with <u>AllDayEvent</u> property
- %4, Indicates the identifier of the event's group, as LONG type, equivalent with <u>GroupID</u> property.
- **%5**, Indicates the caption of the event, as STRING expression, equivalent with <u>Caption</u> property.
- %6, Indicates the extra data associated with the event, as VARIANT type, equivalent with <u>UserData</u> property.
- %7, Gets or sets the duration of the event as FLOAT expression. Above you can find how you can display the duration of the event in hours, minutes...
- %8, Specifies the repetitive expression of the event, equivalent with <u>Repetitive</u> property.
- %256, Gets the margins of the event in a short format, as a STRING expression. The <u>ShortDateFormat</u> property defines the short date format. The <u>ShortTimeFormat</u> property defines the short time format.
- %257, Gets the margins of the event in a long format, as a STRING expression. The <u>LongDateFormat</u> property defines the long date format. The <u>LongTimeFormat</u> property defines the long time format.
- %258, Gets the starting date (not including the time) of the current event, as a DATE type.
- %259, Gets the starting time (not including the date) of the current event, as DATE

type from 0 to 1.

- **%260**, Gets the ending date (not including the time) of the current event, as a DATE type.
- **%261**, Gets the ending time (not including the date) of the current event, as DATE type from 0 to 1.
- %262, Gets the label of the owner group, as STRING expression, equivalent with <u>Caption</u> property of the Group's event.
- %263, Gets the title of the owner group, as STRING expression, equivalent with <u>Title</u> property of the Group's event.
- %264, Indicates if the current event is a repetitive event, as BOOL type. This flag returns TRUE, if the <u>Repetitive</u> property is not empty, and the expression is valid.

The FORMULA, is identified by <%=FORMULA%>, and supports 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: "<i>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:

- \circ 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
- Itrim (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 property supports the following built-in HTML tags:

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

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

- exp, stores the plain text to be shown once the user clicks the anchor, such as "<a ;exp=show lines>"
- e64, encodes in BASE64 the HTML text to be shown once the user clicks the anchor, such as "<a ;e64=gA8ABmABnABjABvABshIAOQAEAAHAAGESikWio+ABzABohp3iELABpABu

" that displays show lines- in gray when the user clicks the + anchor. The "gA8ABmABnABjABvABshIAOQAEAAHAAGESikWio+ABzABohp3iELABpABuABljY string encodes the "<fgcolor 808080>show lines<a>-</fgcolor>" The Decode64Text/Encode64Text methods of the eXPrint can be used to decode/encode e64 fields.

Any ex-HTML caption can be transformed to an expandable-caption, by inserting the anchor ex-HTML tag. For instance, "<solidline>Header</solidline>

Line1<r><a ;exp=show lines>+
Line2
Line3" shows the Header in underlined and bold on the first line and Line1, Line2, Line3 on the rest. The "show lines" is shown instead of Line1, Line2, Line3 once the user clicks the + sign.

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

- **<fgcolor rrggbb>** ... **</fgcolor>** or **<**fgcolor=rrggbb> ... **</**fgcolor> displays text with a specified foreground color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- <bgcolor rrggbb> ... </bgcolor> or <bgcolor=rrggbb> ... </bgcolor> displays text
 with a specified background color. The rr/gg/bb represents the red/green/blue values of
 the color in hexa values.
- **<solidline rrggbb>** ... **</solidline>** or **<**solidline=rrggbb> ... **<**/solidline> draws a solidline 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
- number[:width] 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.
- key[:width] 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 FFFFF;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" generates the following picture:

shadow

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

outline anti-aliasing

property Event.MaxDate as Date

Indicates the max date for the event.

Туре	Description
Date	A DATE expression that specifies the upper limit of the event.

By default, the MaxDate property is 12/31/9999. The MaxDate property indicates the upper limit of the event. In other words, the <u>Start</u> or <u>End</u> properties of the Event can not be greater than the MaxDate property. You can use the <u>MinDate</u>/MaxDate property to limit the area where the event could occur. The <u>Resizable</u> property of the Event indicates whether the user can resize the event at runtime (start, end or both). The <u>Movable</u> property of the Event indicates whether the user can move the event at runtime. You can use the <u>MinDate/MaxDate</u> property of the Calendar object to limit the dates that the calendar can show.

property Event.MinDate as Date

Indicates the min date for the event.

Туре	Description
Date	A DATE expression that specifies the lower limit of the event.

By default, the MinDate property is 1/1/100. The MinDate property indicates the lower limit of the event. In other words, the <u>Start</u> or <u>End</u> properties of the Event can not be less than the MinDate property. You can use the MinDate/<u>MaxDate</u> property to limit the area where the event could occur. The <u>Resizable</u> property of the Event indicates whether the user can resize the event at runtime (start, end or both). The <u>Movable</u> property of the Event indicates whether the user can move the event at runtime. You can use the <u>MinDate/MaxDate</u> property of the Calendar object to limit the dates that the calendar can show.

property Event.Movable as Boolean

Specifies whether the user can move the event.

Туре	Description
Boolean	A boolean expression that specifies whether an event can be moved at runtime.

By default, the Movable property of the Event is True. The Movable property indicates whether the user can move the event at runtime. The <u>MinDate/MaxDate</u> properties of the Event indicates the lower or upper margins where the event can be moved. The <u>UpdateEvent</u> event occurs once an event is resized or moved. The <u>AllowMoveEvent</u> property indicates the combination of the keys to let user moves the events. The <u>AllowMoveEventToOtherGroup</u> property indicates whether an event can be moved from a group to another.

The <u>UpdateEventsLabel</u> property indicates the HTML format to be shown on the label when the user moves the events. The <u>UpdateEventsLabelAlign</u> property aligns the label being shown when the user moves the event. The

Background(exScheduleUpdateEventsBackColor) and

<u>Background</u>(exScheduleUpdateEventsForeColor) specifies the visual appearance of the event being moved. The <u>AllowCreateEvent</u> property indicates the combination of the keys to let user creates new events. The <u>AllowResizeEvent</u> property indicates the combination of the keys to let user resizes the events. The <u>Selectable</u> property specifies whether the event can be selected at runtime. The <u>Resizable</u> property specifies whether the event can be resized at runtime.

For instance, the The <u>AllowMoveEvent</u> property on exDisallow, indicates that no event can be moved at runtime.

method Event.MoveBy (By as Variant)

Moves the event by specified time.

Туре	Description
By as Variant	A double expression that specifies the delay to move the current event, or a string expression that indicates the hour and minutes to delay the current event. For instance, MoveBy("00:15") moves the current event 15 minutes later.

The MoveBy method moves the current event to a new position. The MoveBy method adds the specified delay to <u>Start</u> and <u>End</u> properties of the event. The <u>UpdateEvent</u> event is fired once the margins of the events are updated. You can use the <u>KnownProperty(</u> exEventDuration) to change the event's duration. The MoveBy method can be used to programmatically move the specified event. You can use the <u>MinDate/MaxDate</u> property specifies limit of the event where it should occur.

The following samples move the specified event back with 15 minutes:

VBA (MS Access, Excell...)

With Schedule1
.BeginUpdate
.Calendar.Selection = #5/24/2012#
.Events.Add(#5/24/2012 10:00:00 AM#,#5/24/2012 0:00:00 PM#). MoveBy
"-00:15"
.EndUpdate
End With

VB6

```
With Schedule1

.BeginUpdate

.Calendar.Selection = #5/24/2012#

.Events.Add(#5/24/2012 10:00:00 AM#,#5/24/2012 0:00:00 PM#).MoveBy

"-00:15"

.EndUpdate

End With
```

VB.NET

```
With Exschedule1

.BeginUpdate()

.Calendar.Selection = #5/24/2012#

.Events.Add(#5/24/2012 10:00:00 AM#,#5/24/2012 0:00:00

PM#).MoveBy("-00:15")

.EndUpdate()

End With
```

VB.NET for /COM

```
With AxSchedule1

.BeginUpdate()

.Calendar.Selection = #5/24/2012#

.Events.Add(#5/24/2012 10:00:00 AM#,#5/24/2012 0:00:00

PM#).MoveBy("-00:15")

.EndUpdate()

End With
```

C++

```
Copy and paste the following directives to your header file as
it defines the namespace 'EXSCHEDULELib' for the library: 'ExSchedule 1.0 Control
Library'
```

```
#import <ExSchedule.dll>
using namespace EXSCHEDULELib;
*/
EXSCHEDULELib::ISchedulePtr spSchedule1 = GetDlgItem(IDC_SCHEDULE1)-
>GetControlUnknown();
spSchedule1->BeginUpdate();
spSchedule1->GetCalendar()->PutSelection("5/24/2012");
spSchedule1->GetEvents()->Add("5/24/2012 10:00:00 AM","5/24/2012 12:00:00
PM")->MoveBy("-00:15");
spSchedule1->EndUpdate();
```

```
Schedule1->BeginUpdate();
Schedule1->Calendar->set_Selection(TVariant(TDateTime(2012,5,24).operator
double()));
Schedule1->Events->Add(TVariant(TDateTime(2012,5,24,10,00,00,0).operator
double()),TVariant(TDateTime(2012,5,24,12,00,00,0).operator double()))-
>MoveBy(TVariant("-00:15"));
Schedule1->EndUpdate();
```

C#

```
exschedule1.BeginUpdate();
exschedule1.Calendar.Selection =
Convert.ToDateTime("5/24/2012",System.Globalization.CultureInfo.GetCultureInfo("en-
US"));
exschedule1.Events.Add(Convert.ToDateTime("5/24/2012 10:00:00
AM",System.Globalization.CultureInfo.GetCultureInfo("en-
US")),Convert.ToDateTime("5/24/2012 12:00:00
PM",System.Globalization.CultureInfo.GetCultureInfo("en-US"))).MoveBy("-00:15");
exschedule1.EndUpdate();
```

JavaScript

```
<OBJECT classid="clsid:9B09E13D-7A88-4299-9DBE-383380435377"
id="Schedule1"> </OBJECT>
<SCRIPT LANGUAGE="JScript">
Schedule1.BeginUpdate();
Schedule1.Calendar.Selection = "5/24/2012";
Schedule1.Events.Add("5/24/2012 10:00:00 AM","5/24/2012 12:00:00
PM").MoveBy("-00:15");
Schedule1.EndUpdate();
</SCRIPT>
```

C# for /COM

axSchedule1.BeginUpdate();

```
axSchedule1.Calendar.Selection =
Convert.ToDateTime("5/24/2012",System.Globalization.CultureInfo.GetCultureInfo("en-US"));
axSchedule1.Events.Add(Convert.ToDateTime("5/24/2012 10:00:00
AM",System.Globalization.CultureInfo.GetCultureInfo("en-US")),Convert.ToDateTime("5/24/2012 12:00:00
PM",System.Globalization.CultureInfo.GetCultureInfo("en-US"))).MoveBy("-00:15");
axSchedule1.EndUpdate();
```

X++ (Dynamics Ax 2009)

```
public void init()
{
    COM com_Event;
    anytype var_Event;
    ;
    super();
    exschedule1.BeginUpdate();
exschedule1.Calendar().Selection(COMVariant::createFromDate(str2Date("5/24/2012",2
    var_Event =
    COM::createFromObject(exschedule1.Events()).Add(COMVariant::createFromUtcDateTir
    10:00:00",213)),COMVariant::createFromUtcDateTime(str2Datetime("5/24/2012
    12:00:00",213))); com_Event = var_Event;
    com_Event.MoveBy("-00:15");
    exschedule1.EndUpdate();
}
```

Delphi 8 (.NET only)

```
with AxSchedule1 do
begin
BeginUpdate();
Calendar.Selection := '5/24/2012';
```

```
Events.Add('5/24/2012 10:00:00 AM','5/24/2012 12:00:00 PM').MoveBy('-00:15');
EndUpdate();
end
```

Delphi (standard)

```
with Schedule1 do
begin
    BeginUpdate();
    Calendar.Selection := '5/24/2012';
    Events.Add('5/24/2012 10:00:00 AM','5/24/2012 12:00:00 PM').MoveBy('-00:15');
    EndUpdate();
end
```

VFP

```
with thisform.Schedule1
.BeginUpdate
.Calendar.Selection = {^2012-5-24}
.Events.Add({^2012-5-24 10:00:00},{^2012-5-24 12:00:00}).MoveBy("-00:15")
.EndUpdate
endwith
```

dBASE Plus

```
local oSchedule
```

```
oSchedule = form.Activex1.nativeObject
oSchedule.BeginUpdate()
oSchedule.Calendar.Selection = "05/24/2012"
oSchedule.Events.Add("05/24/2012 10:00:00","05/24/2012
12:00:00").MoveBy("-00:15")
oSchedule.EndUpdate()
```

XBasic (Alpha Five)

Dim oSchedule as P

```
oSchedule = topparent:CONTROL_ACTIVEX1.activex
oSchedule.BeginUpdate()
oSchedule.Calendar.Selection = {05/24/2012}
oSchedule.Events.Add({05/24/2012 10:00:00},{05/24/2012
12:00:00}).MoveBy("-00:15")
oSchedule.EndUpdate()
```

Visual Objects

```
oDCOCX_Exontrol1:BeginUpdate()
oDCOCX_Exontrol1:Calendar:Selection := SToD("20120524")
oDCOCX_Exontrol1:Events:Add(SToD("20120524 10:00:00"),SToD("20120524
12:00:00")):MoveBy("-00:15")
oDCOCX_Exontrol1:EndUpdate()
```

PowerBuilder

OleObject oSchedule

oSchedule = ole_1.Object oSchedule.BeginUpdate() oSchedule.Calendar.Selection = 2012-05-24 oSchedule.Events.Add(DateTime(2012-05-24,10:00:00),DateTime(2012-05-24,12:00:00)).**MoveBy**("-00:15") oSchedule.EndUpdate()

property Event.Pictures as String

Specifies the list of pictures to be displayed on the event.

A string expression that specifies the list of pictures to shown on the event's body. The event's body can displ one ore more pictures at the time, on different lines. F instance: "1,2/pic1" displays the 1 and 2 icons on the f line, while pic2 is displayed on the second line.	o be olay ⁻ or first

By default, the Pictures property is "", which means that initially no pictures are being displayed on the event. The Pictures or/and <u>ExtraPictures</u> property displays a collection of icons, pictures in the event's body. The pictures are shown on the event's body only if they fit the event's body. For instance, if the event is too small, the <u>ShortLabel</u> is displayed, and no icons or pictures are displayed. The <u>PicturesAlign</u> property indicates the alignment of the pictures relative to the event's borders.

The <u>Images</u> method loads icons to the control, <u>HTMLPicture</u> assigns a key to a picture object, and the <u>Pictures</u> collection handles the identifiers of the pictures that can be used in the Pictures or <u>ExtraPictures</u> properties. The <u>AddEvent</u> event notifies your application once a new event is added. You can use this event to initialize the Pictures/PicturesAlign or ExtraPictures/ExtraPicturesAlign properties.

In conclusion, in order to display an icon or a picture in the control you need first to load the icons or the pictures you plan to display, using the <u>Images</u> method, <u>HTMLPicture</u>, or <u>Add</u> method of the ExPictures collection. The Images collection can display only 16x16 icons, while the <u>HTMLPicture</u>, or <u>Add</u> method can load and display custom sized pictures. The <u>Width/Height</u> property specifies the width and height of the picture to be displayed in the event's body.

The event can display icons, pictures several ways as follows:

- Using the HTML tag, if used in any label properties such as LongLabel or ExtraLabel property. The ShortLabel property can not display images or HTML font attributes. If the tag is included in a <a> HTML tag, you have a clickable image through the <u>AnchorClick</u> event. This option can be used to display a default icon or picture for all events in the control using the <u>DefaultEventLongLabel</u> property.
- Using the **Pictures** property of the Event object, which indicates a list of pictures from Pictures collection to be displayed on the event's body. The <u>PicturesAlign</u> property indicates the alignment of the pictures relative to the borders of the event. The hand cursor is shown, if the ExPicture.<u>ShowCursorHand</u> property is set, when the cursor hovers the picture. The <u>PictureClick</u> event is fired if the user clicks a picture on event's picture. You can use the <u>PictureFromPoint</u> property to get the identifier of the picture

from the cursor.

• Using the ExtraPictures property of the Event object, which indicates a list of pictures from Pictures collection to be displayed on the event's body. The ExtraPicturesAlign property indicates the alignment of the pictures relative to the borders of the event. The hand cursor is shown, if the ExPicture.ShowCursorHand property is set, when the cursor hovers the picture. The PictureClick event is fired if the user clicks a picture on event's picture. You can use the PictureFromPoint property to get the identifier of the picture from the cursor.

The Picture and ExtraPictures may display one or more pictures at the time. The , character indicates the separator of pictures in the same line, while the / character divides the lines to show the pictures. For instance, "1,2" displays icon with the index 1 and 2 on the same line, while the "1/2,pic1" displays the first icon on the first line, the second icon and the picture pic1 on the second line.

property Event.PicturesAlign as ContentAlignmentEnum

Indicates the alignment of the event's picture.

Туре	Description
<u>ContentAlignmentEnum</u>	A ContentAlignmentEnum expression that specifies the alignment of the pictures being displayed by the Pictures property.

By default, the PicturesAlign property is exBottomRight, which means any picture associated to the event is displayed on the bottom-right corner of the event. The PicturesAlign property has effect only if the <u>Pictures</u> property refers valid pictures, and the event's body has enough space to display the pictures. The <u>AddEvent</u> event notifies your application once a new event is added. You can use this event to initialize the Pictures/PicturesAlign or ExtraPictures/ExtraPicturesAlign properties. The <u>Width/Height</u> property specifies the width and height of the picture to be displayed in the event's body.

property Event.Repetitive as String

Returns or sets the expression to determine the repetitive event.

Туре	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. If You specify a not empty and valid formula for the Repetitive property, the time part of the <u>Start</u> and <u>End</u> properties determines the time to start and end the repetitive event. The date part is determined by the Repetitive expression. You can use the <u>KnownProperty(</u> exEventRepetitive) property to determine whether the current event is repetitive or not. The property returns True, if the Repetitive property is not empty and valid, and False, if the Repetitive property is empty or not valid.

Starting with the version 12.1, the control supports two ways of representing a Repetitive/Recurrence event:

- <u>Value</u> format, when using the **value** keyword. For instance, "weekday(value) = 1", the event occurs every Monday
- <u>ICalendar</u> format, as described in <u>RFC 5545</u>. 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#, #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
- Itrim (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 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 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 specified frequency. For example, within a MONTHLY rule, FREQ rule part set to YEARLY 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 when the FREQ rule part is not present 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.Resizable as EventResizableEnum

Specifies whether the user can resizes the event at runtime.

Туре

Description

EventResizableEnum	An EventResizableEbum expression that specifies whether
	an event can be resized at runtime.

BY default, the Resizable property is exResizableBoth, which indicates that the starting or ending margins of the event can be resized at runtime. The Resizable property indicates whether an event can be resized or what margin of the event can be resized. The <u>MinDate/MaxDate</u> properties of the Event indicates the lower or upper margins where the event can be moved. The <u>UpdateEvent</u> event occurs once an event is resized or moved. The <u>AllowResizeEvent</u> property indicates the combination of the keys to let user resizes the events. The <u>UpdateEventsLabel</u> property indicates the HTML format to be shown on the label when the user moves the events. The <u>UpdateEventsLabelAlign</u> property aligns the label being shown when the user moves the event. The

Background(exScheduleUpdateEventsBackColor) and

<u>Background</u>(exScheduleUpdateEventsForeColor) specifies the visual appearance of the event being moved. The <u>AllowCreateEvent</u> property indicates the combination of the keys to let user creates new events. The <u>AllowMoveEvent</u> property indicates the combination of the keys to let user moves the events. The <u>Selectable</u> property specifies whether the event can be selected at runtime. The <u>Movable</u> property specifies whether the event at runtime.

For instance, the The <u>AllowMoveEvent</u> property on exDisallow, indicates that no event can be moved at runtime.

property Event.Selectable as Boolean

Specifies whether the user can selects the event.

Туре	Description
Boolean	A boolean expression that specifies whether an event can be selected at runtime.

By default, the Selectable property of the Event is True. The Selectable property indicates whether the user can select the event at runtime. The <u>Selected</u> property of the Event indicates whether the current event is selected or unselected. The <u>AllowSelectEvent</u> property indicates the combination of the keys to let user selects the events. The <u>LayoutStartChanging</u>(exScheduleSelectionChange) event occurs once the selection in the schedule view is about to be changed. The

LayoutEndChanging(exScheduleSelectionChange) event once the selection in the schedule view is changed. The <u>Selection</u> property gets or sets a safe array of selected events. The /NET or /WPF version provides the SelEvents property of List<Event> type to get or sets the new selection using a collection of Event objects. The <u>SelectEventStyle</u> property indicates the way the selected events are shown. The <u>SelectEventColor</u> property specifies the visual appearance of the selected event. The <u>SelectEventTextColor</u> property specifies the foreground color of the selected event.

The <u>AllowCreateEvent</u> property indicates the combination of the keys to let user creates new events. The <u>AllowResizeEvent</u> property indicates the combination of the keys to let user resizes the events. The <u>Movable</u> property specifies whether the event can be moved at runtime. The <u>Resizable</u> property specifies whether the event can be resized at runtime.

For instance, the The <u>AllowSelectEvent</u> property on exDisallow, indicates that no event can be moved at runtime.

property Event.Selected as Boolean

Selects or unselects the current event.

Туре	Description
Boolean	A boolean expression that specifies whether an event is selected or unselected.

By default, the Selected property of the Event is False. The Selected property indicates whether the event is selected or unselected. The <u>Selectable</u> property of the Event indicates whether the event can be selected at runtime. The <u>AllowSelectEvent</u> property indicates the combination of the keys to let user selects the events. The <u>LayoutStartChanging(</u> exScheduleSelectionChange) event occurs once the selection in the schedule view is about to be changed. The <u>LayoutEndChanging(</u> exScheduleSelectionChange) event once the selection in the schedule view is changed. The <u>Selected</u> events as a safe array of selected events. The /NET or /WPF version provides the SelEvents property of List<Event> type to get or sets the new selection using a collection of Event objects. The <u>SelectEventStyle</u> property indicates the way the selected events are shown. The <u>SelectEventColor</u> property specifies the visual appearance of the selected event. The <u>SelectEventTextColor</u> property specifies the foreground color of the selected event.

The <u>AllowCreateEvent</u> property indicates the combination of the keys to let user creates new events. The <u>AllowResizeEvent</u> property indicates the combination of the keys to let user resizes the events. The <u>Movable</u> property specifies whether the event can be moved at runtime. The <u>Resizable</u> property specifies whether the event can be resized at runtime.

For instance, the The <u>AllowSelectEvent</u> property on exDisallow, indicates that no event can be moved at runtime.

property Event.ShortLabel as String

Specifies the short label to be displayed on the event.

Туре	Description
String	A string expression that specifies the extended HTML label, to be displayed on the event's body. <i>The images, or any font HTML attribute is ignored.</i>

By default, the ShortLabel property is initialized with the value of the <u>DefaultEventShortLabel</u> property. The ShortLabel property is shown if the event's body is too small. The <u>LongLabel</u> property is displayed only if it fits the event's body, else the ShortLabel property is shown. The event displays the <u>ExtraLabel</u>, only if the LongLabel property is displayed. The <u>AddEvent</u> event occurs once a new event is added to the Events collection. The <u>LabelAlign</u> property specifies the alignment of the long label. The <u>DefaultEventPadding</u> property indicates the padding of the labels on the event, relative to event's borders. *The ShortLabel property displays (ignores) NO images such as , or font HTML attributes such as , <i>, … The KnowProperty(exEventShortLabel) is equivalent with the ShortLabel property.*

Here's a few samples:

- "new", simple new text is shown.
- "<%=%256%>", displays the event's start and end points in a short format.
- "<%=%257%>", displays the event's margins in a long format.
- "Start: <%=%1%>
End: <%=%2%>", displays the starting margin of the even on the first line, while on the second line it displays the ending point of the event.
- "<%=%256%>
Caption: <%=%5%>", displays the event's margins in short format on the first line, and on the second line it displays the event's <u>Caption</u> property. The caption shown on the event's body is automatically updated once the event is moved to a new position or the event's Caption is changed.
- "<%=%256%>
<%=%264? `repetitive event`:``%>" displays automatically the "repetitive event" for repetitive events, or when the event's <u>Repetitive</u> property is not empty and valid
- "Duration: <%=((1:=int(0:= (date(%2)-date(%1)))) != 0 ? (=:1 + ' day(s)') : ") + (=:1 ? ' ' : ") + ((1:=int(0:=((=:0 =:1 + 1/24/60/60/2)*24))) != 0 ? =:1 + ' hour(s)' : ") + (=:1 ? ' ' : ") + ((1:=round((=:0 =:1)*60)) != 0 ? =:1 + ' min(s)' : ")%>" displays the duration of the event in days, have and minutes.

of the event in days, hours and minutes.

"<%=%256%>
Duration: <%=((1:=int(0:= (date(%2)-date(%1)))) != 0 ? (=:1 + ' day(s)') : ") + (=:1 ? ' ' : ") + ((1:=int(0:=((=:0 - =:1 + 1/24/60/60/2)*24))) != 0 ? =:1 + ' hour(s)' : ") + (=:1 ? ' ' : ") + ((1:=round((=:0 - =:1)*60)) != 0 ? =:1 + ' min(s)' : ")%>" displays the event's margins on the first line and the duration of the event in days, hours and minutes, on the second line

"<%=%><%=%5%>
<<%=%256%>", displays the event's Caption on first line(s), following by the event's Start/End margins in short date-time format. The <%=%> prefix forces the expression to be re-evaluated and apply any HTML tag found. For instance, %5 indicates the event's Caption property, and if it contains HTML tags they will be applied as is, instead displaying them as a plain text. Any expression that starts with "<%=%>" is re- evaluated and its result is displayed in HTML format (available starting with the version 12.2)

The <u>EventKnowPropertyEnum</u> defines the %identifiers that can be used in formula <%=FORMULA%>. For instance, the ShortLabel property on "Start: <%=time(%1) replace `AM` with ``%>" displays the time when the event starts with no AM time indicators.

The property supports the following identifiers. These identifiers can be used in FORMULA format:

- %1, Indicates the starting date/time of the event as DATE type, equivalent with <u>Start</u> property
- %2, Indicates the ending date/time of the event as DATE type, equivalent with End property
- %3, Indicates if the current event is an all day event as BOOL type, equivalent with <u>AllDayEvent</u> property
- %4, Indicates the identifier of the event's group, as LONG type, equivalent with <u>GroupID</u> property.
- **%5**, Indicates the caption of the event, as STRING expression, equivalent with <u>Caption</u> property.
- %6, Indicates the extra data associated with the event, as VARIANT type, equivalent with <u>UserData</u> property.
- %7, Gets or sets the duration of the event as FLOAT expression. Above you can find how you can display the duration of the event in hours, minutes...
- %8, Specifies the repetitive expression of the event, equivalent with <u>Repetitive</u> property.
- %256, Gets the margins of the event in a short format, as a STRING expression. The <u>ShortDateFormat</u> property defines the short date format. The <u>ShortTimeFormat</u> property defines the short time format.
- %257, Gets the margins of the event in a long format, as a STRING expression. The <u>LongDateFormat</u> property defines the long date format. The <u>LongTimeFormat</u> property defines the long time format.
- %258, Gets the starting date (not including the time) of the current event, as a DATE type.
- %259, Gets the starting time (not including the date) of the current event, as DATE type from 0 to 1.
- %260, Gets the ending date (not including the time) of the current event, as a DATE type.

- **%261**, Gets the ending time (not including the date) of the current event, as DATE type from 0 to 1.
- %262, Gets the label of the owner group, as STRING expression, equivalent with <u>Caption</u> property of the Group's event.
- %263, Gets the title of the owner group, as STRING expression, equivalent with <u>Title</u> property of the Group's event.
- %264, Indicates if the current event is a repetitive event, as BOOL type. This flag returns TRUE, if the <u>Repetitive</u> property is not empty, and the expression is valid.

The FORMULA, is identified by <%=FORMULA%>, and supports 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#, #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
- Itrim (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)

property Event.ShowStatus as Boolean

Specifies whether the current event shows or hides its status.

Туре	Description
Boolean	A Boolean expression that specifies whether the event shows its status part.

By default, the ShowStatus property is True. The ShowStatus property can be used to show or hide the status part of a specified event. The <u>ClearShowStatus</u> method clears the ShowStatus flag, if previously the ShowStatus property has been set. By default, the <u>ShowStatusEvent</u> property shows or hides the status part for all events. The <u>ShowStatus</u> property shows or hides the status part of giving event. Use the ClearShowStatus method to allow the <u>ShowStatusEvent</u> property to display the event's status, rather than ShowStatus property.

You can:

- show the status part for all events (<u>ShowStatusEvent</u> on True), and use the ShowStatus (ShowStatus on False) property to hide the status part for specified events only.
- hide the status part for all events (<u>ShowStatusEvent</u> on False), and use the ShowStatus (ShowStatus on True) property to show the status part for specified events only.

property Event.Start as Date

Specifies the starting date/time of the event.

Туре	Description
Date	A DATE expression that specifies the lower margin of the event

The Start property of the Event indicates the date/time when the event or the appointment begins. The Start parameter of the Add method indicates the starting point of the event at adding time. The End property of the event indicates the ending point of the event. The Start and End properties may be identical if the <u>AllDayEvent</u> property is True. The <u>UpdateEvent</u> event is fired once the Start property is changed. The <u>Resizable</u> property of the Event indicates whether the user can resize the event at runtime (start, end or both). The <u>Movable</u> property of the Event indicates whether the user can move the event at runtime. You can use the <u>MinDate/MaxDate</u> property specifies the range of dates where the Start/End can be shown. You can use the <u>MoveBy</u> method to delay the current event with a specified value time. You can use the <u>KnownProperty</u>(exEventDuration) to change the event's duration.

The <u>KnownProperty</u>(exEventEndDateTime) property indicates the End property on a label property such as: <u>DefaultEventLongLabel</u>, <u>DefaultEventShortLabel</u>, <u>CreateEventLabel</u>, <u>UpdateEventsLabel</u>, <u>ShortLabel</u>, <u>LongLabel</u> and <u>ExtraLabel</u>.

- You can use the <u>KnownProperty</u>(exEventStartDate)/<u>KnownProperty</u>(exEventStartTime) property to extract the starting date/time of the event.
- You can use the <u>KnownProperty</u>(exEventEndDate)/<u>KnownProperty</u>(exEventEndTime) property to extract the ending date/time of the event.
- You can use the <u>KnownProperty</u>(exEventDuration) property to specifies the duration/length of the event.

property Event.StartUpdateEvent as Long

Starts changing properties of the calendar-event, so EndUpdateEvent method adds programmatically updated properties to undo/redo queue.

Туре	Description
Long	A Long expression that specifies the handle to be passed to <u>EndUpdateEvent</u> so the updated properties of the bar are added to the Undo/Redo queue of the chart, so they can be used in undo/redo operations.

The StartUpdateEvent/EndUpdateEvent methods record and add changes of the current calendar-event to the control's Undo/Redo queue. You can use the <u>StartBlockUndoRedo</u> / <u>EndBlockUndoRedo</u> methods to group multiple Undo/Redo operations into a single-block. The <u>AllowUndoRedo</u> property specifies whether the control supports undo/redo operations for objects (calendar-events). No entry is added to the Undo/Redo queue if no property is changed for the current calendar-event. Each call of the StartUpdateEvent must be succeeded by a <u>EndUpdateEvent</u> call. The <u>UndoListAction</u> property lists the Undo actions that can be performed in the chart. The <u>RedoListAction</u> property lists the Redo actions that can be performed in the chart.

The StartUpdateEvent/<u>EndUpdateEvent</u> methods can record changes for all properties from 1 (exEventStartDateTime) to EXEVENTMAX listed by <u>EventKnownPropertyEnum</u> type.

The Undo/Redo records show as:

 "UpdateEvent; EVENTID", indicates that one or more properties of the calendar-event has been updated, using the StartUpdateEvent / <u>EndUpdateEvent</u> methods

within the <u>UndoListAction</u>/<u>RedoListAction</u> result.

property Event.StatusColor as Color

Specifies the color of the event's status.

Туре	Description
Color	A Color expression that specifies the background color to show the status part of the event. The last 7 bits in the high significant byte of the color to indicates the identifier of the skin being used. Use the <u>Add</u> 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.

By default, the StatusColor property is 0. By default, <u>StatusEventColor</u> property indicates the color to show the event's status, while the StatusColor property is zero. The StatusColor property indicates the color to show the status part of a specified event. The <u>StatusEventColor</u> property indicates the color to show the status part of the events. The <u>StatusEventSize</u> property specify the size in pixels of the event's status.

You can:

- show the status part for all events (<u>ShowStatusEvent</u> on True), and use the ShowStatus (<u>ShowStatus</u> on False) property to hide the status part for specified events only.
- hide the status part for all events (<u>ShowStatusEvent</u> on False), and use the ShowStatus (<u>ShowStatus</u> on True) property to show the status part for specified events only.

property Event.StatusPattern as Pattern

Specifies the pattern of the event (status)

Туре	Description
Pattern	A Pattern object associated with the event's status.

By default, the StatusPattern.Type property exPatternEmpty which indicates that no pattern is shown, on the event's status. The <u>Color</u> property indicates the color to display the pattern. The <u>FrameColor</u> property indicates the color to show the frame, if the exPatternFrame flag is included in the Type property. The <u>EventPattern</u> property indicates the pattern to be shown when events belongs to different groups. The <u>BodyPattern</u> property specifies the pattern to be shown on the event's body. The <u>ShowStatusEvent</u> property shows or hides the status part of all events. The <u>ShowStatus</u> property shows or hides the status part of all events.

property Event.ToolTip as String

Indicates the tooltip to be shown when the cursor hovers the event.

Туре	Description
String	A String expression that defines the extended HTML format to be displayed when the cursor hovers the appointments.

By default, the ToolTip property is initialized with the value of the <u>DefaultEventTooltip</u> property. The <u>ShowToolTip</u> method can be used during the <u>MouseMove</u> event to display a custom tooltip. The <u>ToolTipTitle</u> property defines the title of the event's tootip. Use the <u>ToolTipPopDelay</u> property specifies the period in ms of time the ToolTip remains visible if the mouse pointer is stationary within a control. Use the <u>ToolTipFont</u> property to change the tooltip's font.

Here's a few samples:

l

- "new", simple new text is shown.
- "<%=%256%>", displays the event's start and end points in a short format.
- "<%=%257%>", displays the event's margins in a long format.
- "Start: <%=%1%>
End: <%=%2%>", displays the starting margin of the even on the first line, while on the second line it displays the ending point of the event.
- "<%=%256%>
Caption: <%=%5%>", displays the event's margins in short format on the first line, and on the second line it displays the event's <u>Caption</u> property. The caption shown on the event's body is automatically updated once the event is moved to a new position or the event's Caption is changed.
- "<%=%256%>
<%=%264? `repetitive event`:``%>" displays automatically the "repetitive event" for repetitive events, or when the event's <u>Repetitive</u> property is not empty and valid
- "Duration: <%=((1:=int(0:= (date(%2)-date(%1)))) != 0 ? (=:1 + ' day(s)') : ") + (=:1 ? ' ' : ") + ((1:=int(0:=((=:0 =:1 + 1/24/60/60/2)*24))) != 0 ? =:1 + ' hour(s)' : ") + (=:1 ? ' ' : ") + ((1:=round((=:0 =:1)*60)) != 0 ? =:1 + ' min(s)' : ")%>" displays the duration of the event in days, hours and minutes.
- "<%=%256%>
Duration: <%=((1:=int(0:= (date(%2)-date(%1)))) != 0 ? (=:1 + ' day(s)') : ") + (=:1 ? ' ' : ") + ((1:=int(0:=((=:0 =:1 + 1/24/60/60/2)*24))) != 0 ? =:1 + ' hour(s)' : ") + (=:1 ? ' ' : ") + ((1:=round((=:0 =:1)*60)) != 0 ? =:1 + ' min(s)' : ")%>" displays the event's margins on the first line and the duration of the event in days, hours and minutes, on the second line

The <u>EventKnowPropertyEnum</u> defines the %identifiers that can be used in formula <%=FORMULA%>. For instance, the ToolTip property on "Start: <%=time(%1) replace `AM` with ``%>" displays the time when the event starts with no AM time indicators.

The property supports the following identifiers. These identifiers can be used in FORMULA format:

- **%1**, Indicates the starting date/time of the event as DATE type, equivalent with <u>Start</u> property
- %2, Indicates the ending date/time of the event as DATE type, equivalent with End property
- %3, Indicates if the current event is an all day event as BOOL type, equivalent with <u>AllDayEvent</u> property
- %4, Indicates the identifier of the event's group, as LONG type, equivalent with <u>GroupID</u> property.
- %5, Indicates the caption of the event, as STRING expression, equivalent with <u>Caption</u> property.
- %6, Indicates the extra data associated with the event, as VARIANT type, equivalent with <u>UserData</u> property.
- %7, Gets or sets the duration of the event as FLOAT expression. Above you can find how you can display the duration of the event in hours, minutes...
- %8, Specifies the repetitive expression of the event, equivalent with <u>Repetitive</u> property.
- %256, Gets the margins of the event in a short format, as a STRING expression. The <u>ShortDateFormat</u> property defines the short date format. The <u>ShortTimeFormat</u> property defines the short time format.
- %257, Gets the margins of the event in a long format, as a STRING expression. The <u>LongDateFormat</u> property defines the long date format. The <u>LongTimeFormat</u> property defines the long time format.
- **%258**, Gets the starting date (not including the time) of the current event, as a DATE type.
- %259, Gets the starting time (not including the date) of the current event, as DATE type from 0 to 1.
- **%260**, Gets the ending date (not including the time) of the current event, as a DATE type.
- %261, Gets the ending time (not including the date) of the current event, as DATE type from 0 to 1.
- %262, Gets the label of the owner group, as STRING expression, equivalent with <u>Caption</u> property of the Group's event.
- %263, Gets the title of the owner group, as STRING expression, equivalent with <u>Title</u> property of the Group's event.
- %264, Indicates if the current event is a repetitive event, as BOOL type. This flag returns TRUE, if the <u>Repetitive</u> property is not empty, and the expression is valid.

The FORMULA, is identified by <%=FORMULA%>, and supports 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#, #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
 - $\circ\,$ #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
- \circ 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
- Itrim (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 property supports the following built-in HTML tags:

- ... displays the text in **bold**
- <i> ... </i> displays the text in *italics*
- <u> ... </u> <u>underlines</u> the text
- <s> ... </s> Strike-through text
- <a id;options> ... displays an <u>anchor</u> element that can be clicked. An anchor is a piece of text or some other object (for example an image) which marks the beginning and/or the end of a hypertext link. The <a> element is used to mark that piece of text (or inline image), and to give its hypertextual relationship to other documents. The control fires the AnchorClick(AnchorID, Options) event when the user clicks the anchor element. The FormatAnchor property customizes the visual effect for anchor elements.
The control supports expandable HTML captions feature which allows you to expand(show)/collapse(hide) different information using <a ;exp=> or <a ;e64=> anchor tags. The exp/e64 field of the anchor stores the HTML line/lines to show once the user clicks/collapses/expands the caption.

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

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

- ... displays portions of text with a different font and/or different size. For instance, the "bit" draws the bit text using the Tahoma font, on size 12 pt. If the name of the font is missing, and instead size is present, the current font is used with a different size. For instance, "bit" displays the bit text using the current font, but with a different size.
- **<fgcolor rrggbb>** ... **</fgcolor>** or **<**fgcolor=rrggbb> ... **</**fgcolor> displays text with a specified foreground color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- <bgcolor rrggbb> ... </bgcolor> or <bgcolor=rrggbb> ... </bgcolor> displays text with a specified background color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- <solidline rrggbb> ... </solidline> or <solidline=rrggbb> ... </solidline> draws a solidline 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
- number[:width] 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.
- key[:width] 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 FFFFF;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=FFFFF>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.ToolTipTitle as String

Indicates the title of the tooltip to be shown when the cursor hovers the event.

Туре	Description
String	A String expression that defines the title of the event's tooltip

By default, the The ToolTipTitle property is "", which means that no title is associated with the tooltip of the event. The <u>ToolTip</u> property indicates the tooltip to be shown when the cursor hovers the event. The ToolTipTitle property defines the title of the event's tootip. The <u>ShowToolTip</u> method can be used during the <u>MouseMove</u> event to display a custom tooltip.

property Event.UserData as Variant

Indicates any extra data associated with the Event object.

Туре	Description
Variant	A VARIANT expression that specifies any extra data associated with the event

By default, the UserData property is empty. You can use the UserData property to associate any extra data to the current event. You can display the event's UserData as string on the body of the event, if the ShortLabel, LongLabel, ExtraLabel or ToolTip properties of the Event includes the <%=%6%> TAG. The RemoveEvent event notifies your application once an event is removed from the Events collection. You can use the RemoveEvent event to release any extra data that has been allocated to the event during creation. The UserData property of the Event, or for any other object is provided to let you associate any extra data to the object. For instance, the UserData property can hold any event related data that is not displayed, like a primary key of an event in database.

Events object

The Events collection holds the control's events/appointments. The Events collection can be accessed through the <u>Events</u> property of the control. The <u>AddEvent</u> event occurs once a new event is added to the Events collection. The <u>RemoveEvent</u> event occurs once an event is removed. The <u>UpdateEvent</u> event occurs once the margins of the event are updated.

The following screen shot shows the schedule view, which displaying the Events of the control:



The Events collection supports the following properties and methods:

Name	Description
Add	Adds an Event object to the collection and returns a reference to the newly created object.
<u>Clear</u>	Removes all objects in a collection.
<u>Count</u>	Returns the number of objects in a collection.
<u>ltem</u>	Returns a specific Event of the Events collection, giving its handle.
Remove	Removes a specific member from the Events collection, giving its handle or reference.

method Events.Add (Start as Variant, End as Variant)

Adds an Event object to the collection and returns a reference to the newly created object.

Туре	Description
Start as Variant	A DATE expressions that specifies the starting point of the event.
End as Variant	A DATE expressions that specifies the ending point of the event.
Return	Description
<u>Event</u>	An Event object being created.

The Add method adds programmatically a new event/appointment to the control. The <u>AddEvent</u> event occurs once a new event is added to the Events collection. The <u>UpdateEvent</u> event occurs once the margins of the event are updated. The <u>LayoutStartCreating</u>(exScheduleCreateEvent) event occurs once a new event is creating using the mouse. The <u>LayoutEndCreating</u>(exScheduleCreateEvent) event occurs once the event has been created at runtime using the mouse. The <u>Start/End</u> properties of the Event indicate the margins of the appointment. The <u>Repetitive</u> property indicates the expression that shows the same event repeatedly in different dates. The <u>Selection</u> property of the Calendar object indicates the date being browsed in the schedule view. Use the <u>Remove</u> method to remove programmatically an event from the control.

The <u>AllowCreateEvent</u> property indicates the combination of keys that user can use to create a new event at runtime. The <u>CreateEventLabel</u> property specifies the label to be shown when the user creates a new event. The <u>CreateEventLabelAlign</u> property indicates the alignment of the label while user creates new events. The <u>Background</u>(exScheduleCreateEventBackColor) and <u>Background</u>(exScheduleCreateEventForeColor) properties indicate the visual aspect of the label being shown to create new events.

The following VB sample combines the AddEvent, LayoutStartCreating(exScheduleCreateEvent) and LayoutEndCreating(exScheduleCreateEvent), to ask the user if he wants to keep the newly created event, and if not, removes it:

Dim iCreatingEvent As Long

```
Private Sub Schedule1_AddEvent(ByVal Ev As EXSCHEDULELibCtl.IEvent)
If Not (iCreatingEvent = 0) Then
If Not MsgBox("Do you allow creating this new event?", vbQuestion Or
```

```
vbYesNoCancel) = vbYes Then
      Schedule1.Events.Remove (Ev.Handle)
    End If
  End If
End Sub
Private Sub Schedule1_LayoutStartChanging(ByVal Operation As
EXSCHEDULELibCtl.LayoutChangingEnum)
  If (Operation = exScheduleCreateEvent) Then
    iCreatingEvent = iCreatingEvent + 1
  End If
End Sub
Private Sub Schedule1_LayoutEndChanging(ByVal Operation As
EXSCHEDULELibCtl.LayoutChangingEnum)
  If (Operation = exScheduleCreateEvent) Then
    iCreatingEvent = iCreatingEvent - 1
  End If
End Sub
```

method Events.Clear ()

Removes all objects in a collection.

Туре

Description

The Clear method of the Events collection clears the events collection. In other words, calling the Clear method erases all elements in the Events collection. The control fires the RemoveEvent each time an event is removed, including when all events are removed, so the <u>RemoveEvent</u> event is fired for each event to be removed. The <u>Remove</u> method removes the specified event. The <u>RemoveSelection</u> method removes or erases all selected events in the schedule view. The <u>ClearAll</u> method clear all objects in the control, including the events. Any of these methods invoke calling of the RemoveEvent event. Use the <u>GroupID</u> property of the Event object to move a (remove/add) an Event from a Group to another. You can use the Events.Clear method to remove any previously event from the scheduler before calling the <u>LoadXML</u> method, which does not remove any event before. You can use the <u>ShowEvents</u> property to indicates whether the regular/repetitive events are shown in the view.

property Events.Count as Long

Returns the number of objects in a collection.

Туре	Description
Long	A long expression that specifies the number of events in the schedule.

The Count property gets the number of Event objects in the Events collection. The <u>Add</u> method of the Events collection adds a new event/appointment to the eXSchedule control, so the Count property is increased. The <u>Clear</u> method clears all events in the collection, and so the Count property is set on 0. The <u>Item</u> property access an individual element of the collection. You can use the <u>Remove</u> method to remove a specific event, so the Count property is decreased.

In order to enumerate the events we recommend using the for each statement, instead for i - 0 to Count - 1 as in the following samples.

The following VB sample shows how you can enumerate all events in the control:

Dim e As EXSCHEDULELibCtl.Event For Each e In Schedule1.Events Debug.Print "Event: " & e.Start & " to " & e.End Next

The following VB/NET sample shows how you can enumerate all events in the control:

For Each ev As exontrol.EXSCHEDULELib.Event In Exschedule1.Events Debug.Print("Event: " & ev.Start.ToString() & " " & ev.End.ToString()) Next

The following C# sample shows how you can enumerate all events in the control:

foreach (exontrol.EXSCHEDULELib.Event ev in exschedule1.Events) System.Diagnostics.Debug.Print("Event: " + ev.Start.ToString() + " " + ev.End.ToString());

The following VFP sample shows how you can enumerate all events in the control:

```
*** ActiveX Control Event ***
LPARAMETERS operation
```

* 1 ' exCalendarSelectionChange If Operation = 1 Then

```
local d
For Each d In thisform.Schedule1.Calendar.Selection
WAIT WINDOW "Select: " + TTOC(d)
ENDFOR
EndIf
```

The following C++ sample shows how you can enumerate all events in the control:

```
IEnumVARIANTPtr spEnum = m_spSchedule->Events->_NewEnum;
if (spEnum != NULL)
{
  spEnum->Reset();
  unsigned long n = 0;
  _variant_t vtElement;
  while(SUCCEEDED(spEnum->Next(1, &vtElement, &n)) && (n!=0))
  {
    EXSCHEDULELib::IEventPtr spEvent = V_DISPATCH( &vtElement );
    if (spEvent != NULL)
    {
      CString sMessage;
      sMessage.Format(_T("Event: %f %f\r\n"), spEvent->Start, spEvent->End );
      OutputDebugString( sMessage );
    }
  }
```

where m_spSchedule is of EXSCHEDULELib::ISchedulePtr type.

property Events.Item (Handle as Variant) as Event

Returns a specific Event of the Events collection, giving its handle.

Туре	Description
Handle as Variant	 A long expression that indicates the handle of the event. The Handle identifies unique the event, and is is allocated by the control. The <u>Handle</u> property of the event specifies the event's handle. A double/float expression that specifies the event's identifier. The <u>KnownProperty</u>(exEventID) property specifies the event's identifier. The Handle is automatically generated by the control, and can not be changed, while the IDentifier can be set by the user.
Event	An Event object associated with the giving Handle.

The Item property access an individual element of the collection. The <u>Count</u> property gets the number of Event objects in the Events collection. The <u>Add</u> method of the Events collection adds a new event/appointment to the eXSchedule control, so the Count property is increased. The <u>Clear</u> method clears all events in the collection, and so the Count property is set on 0. You can use the <u>Remove</u> method to remove a specific event, so the Count property property is decreased.

The following VB sample accesses/prints the Event's user data giving its handle:

With Schedule1.Events Debug.Print .Item(h).UserData End With

The following VB sample accesses/prints the Event's user data giving its identifier:

With Schedule1.Events Debug.Print .Item(CDbl(i)).UserData End With

where the **h** and **i** variables are of long type as in the following statement:

Dim **h** As Long Dim **i** As Long

```
Private Sub Form_Load()

i = 100

With Schedule1.Events.Add(#4/5/2014 10:00:00 AM#, #4/5/2014 2:00:00 PM#)

.KnownProperty(exEventID) = i

.UserData = "this is a bit of text associated with the event"

h = .Handle

End With

End Sub
```

Shortly, the **h** member is generated by the control in the <u>Handle</u> property (read-only), while **i** member can be set by the user.

In order to enumerate the events we recommend using the for each statement, instead for i - 0 to Count - 1 as in the following samples.

The following VB sample shows how you can enumerate all events in the control:

Dim e As EXSCHEDULELibCtl.Event For Each e In Schedule1.Events Debug.Print "Event: " & e.Start & " to " & e.End Next

The following VB/NET sample shows how you can enumerate all events in the control:

For Each ev As exontrol.EXSCHEDULELib.Event In Exschedule1.Events Debug.Print("Event: " & ev.Start.ToString() & " " & ev.End.ToString()) Next

The following C# sample shows how you can enumerate all events in the control:

```
foreach (exontrol.EXSCHEDULELib.Event ev in exschedule1.Events)
System.Diagnostics.Debug.Print("Event: " + ev.Start.ToString() + " " + ev.End.ToString());
```

The following VFP sample shows how you can enumerate all events in the control:

```
For Each e as Object In thisform.Schedule1.Events
LOCAL ee as Object
ee = thisform.Schedule1.Events(e)
WAIT WINDOW"Event " + TTOC(ee.Start) + " " + TTOC(ee.End)
```

ENDFOR

The following C++ sample shows how you can enumerate all events in the control:

```
IEnumVARIANTPtr spEnum = m_spSchedule->Events->_NewEnum;
if (spEnum != NULL)
{
  spEnum->Reset();
  unsigned long n = 0;
  _variant_t vtElement;
  while(SUCCEEDED(spEnum->Next(1, &vtElement, &n)) && (n!=0))
  {
    EXSCHEDULELib::IEventPtr spEvent = V_DISPATCH( &vtElement );
    if (spEvent != NULL)
    {
      CString sMessage;
      sMessage.Format(_T("Event: %f %f\r\n"), spEvent->Start, spEvent->End );
      OutputDebugString( sMessage );
    }
  }
```

where m_spSchedule is of EXSCHEDULELib::ISchedulePtr type.

method Events.Remove (Handle as Variant)

Removes a specific member from the Events collection, giving its handle or reference.

Туре	Description
Handle as Variant	A long expression that indicates the handle of the event. The Handle identifies unique the event, and is is allocated by the control. The <u>Handle</u> property of the event specifies the event's handle. A double/float expression that specifies the event's identifier. The <u>KnownProperty(exEventID)</u> property specifies the event's identifier. The Handle is automatically generated by the control, and can not be changed, while the IDentifier can be set by the user.

The Remove method removes the specified event. The <u>Handle</u> property of the Event object indicates the handle of the event to be removed. The Handle is automatically generated to be unique for any event, and it can not be changed. In other words, the Handle indicates the key to specify an event. You can use the <u>UserData</u> property of the Event to associate any extra data with your event. *If the Remove method is called during the <u>AddEvent</u> event, use the <u>Ev.Handle to remove the event</u>. The control fires the <u>Remove</u> event once an Event is being removed. You can use the Remove any extra data associated with the event. The user can create new events using the mouse, if the <u>AllowCreateEvent</u> is not zero, using the Events.<u>Add</u> method or loading a XML document using the <u>LoadXML</u> method. The <u>RemoveSelection</u> method removes or erases all selected events in the schedule view. The <u>Clear</u> method of the Events collection clears all events in the schedule component. The <u>ClearAll</u> method clear all objects in the control, including the events. Use the <u>GroupID</u> property of the Event object to move a (remove/add) an Event from a Group to another.*

The following VB sample asks the user if he wants to keep the newly created event, and if not, removes it:

Dim iCreatingEvent As Long

```
Private Sub Schedule1_AddEvent(ByVal Ev As EXSCHEDULELibCtl.IEvent)

If Not (iCreatingEvent = 0) Then

If Not MsgBox("Do you allow creating this new event?", vbQuestion Or

vbYesNoCancel) = vbYes Then

Schedule1.Events.Remove (Ev.Handle)

End If

End If

End Sub
```

```
Private Sub Schedule1_LayoutStartChanging(ByVal Operation As
EXSCHEDULELibCtl.LayoutChangingEnum)
If (Operation = exScheduleCreateEvent) Then
iCreatingEvent = iCreatingEvent + 1
End If
End Sub
Private Sub Schedule1_LayoutEndChanging(ByVal Operation As
EXSCHEDULELibCtl.LayoutChangingEnum)
If (Operation = exScheduleCreateEvent) Then
iCreatingEvent = iCreatingEvent - 1
End If
End Sub
```

ExDataObject object

The <u>OleDragDrop</u> event notifies your application that the user drags some data on the control. Defines the object that contains OLE drag and drop information. The ExDataObject object supports the following method and properties:

Name	Description
<u>Clear</u>	Deletes the contents of the ExDataObject object.
<u>Files</u>	Returns an ExDataObjectFiles collection, which in turn contains a list of all filenames used by an ExDataObject object.
<u>GetData</u>	Returns data from an ExDataObject object in the form of a variant.
<u>GetFormat</u>	Returns a value indicating whether an item in the ExDataObject object matches a specified format.
<u>SetData</u>	Inserts data into an ExDataObject object using the specified data format.

method ExDataObject.Clear ()

Deletes the contents of the DataObject object.

Туре

Description

The Clear method can be called only for drag sources. The <u>OleDragDrop</u> event notifies your application that the user drags some data on the control.

property ExDataObject.Files as ExDataObjectFiles

Returns a DataObjectFiles collection, which in turn contains a list of all filenames used by a DataObject object.

Туре	Description
ExDataObjectFiles	An ExDataObjectFiles object that contains a list of filenames used in OLE drag and drop operations.

The Files property is valid only if the format of the clipboard data is exCFFiles. The <u>OleDragDrop</u> event notifies your application that the user drags some data on the control.

method ExDataObject.GetData (Format as Integer)

Returns data from a DataObject object in the form of a variant.

Туре	Description
Format as Integer	An <u>exClipboardFormatEnum</u> expression that defines the data's format
Return	Description
Variant	A Variant value that contains the ExDataObject's data in the given format

Use GetData property to retrieve the clipboard's data that has been dragged to the control. It's possible for the GetData and <u>SetData</u> methods to use data formats other than <u>exClipboardFormatEnum</u>, including user-defined formats registered with Windows via the RegisterClipboardFormat() API function. The GetData method always returns data in a byte array when it is in a format that it is not recognized. Use the <u>Files</u> property to retrieves the filenames if the format of data is exCFFiles

method ExDataObject.GetFormat (Format as Integer)

Returns a value indicating whether the ExDataObject's data is of specified format.

Туре	Description
Format as Integer	A constant or value that specifies a clipboard data format like described in <u>exClipboardFormatEnum</u> enum.
Return	Description
Boolean	A boolean value that indicates whether the ExDataObject's data is of specified format.

Use the GetFormat property to verify if the ExDataObject's data is of a specified clipboard format. The GetFormat property retrieves True, if the ExDataObject's data format matches the given data format.

method ExDataObject.SetData ([Value as Variant], [Format as Variant])

Inserts data into a ExDataObject object using the specified data format.

Туре	Description
Value as Variant	A data that is going to be inserted to ExDataObject object.
Format as Variant	A constant or value that specifies the data format, as described in <u>exClipboardFormatEnum</u> enum

Use SetData property to insert data for OLE drag and drop operations. Use the <u>Files</u> property is you are going to add new files to the clipboard data. The <u>OleDragDrop</u> event notifies your application that the user drags some data on the control.

ExDataObjectFiles object

The ExDataObjectFiles contains a collection of filenames. The ExDataObjectFiles object is used in OLE Drag and drop events. In order to get the list of files used in drag and drop operations you have to use the <u>Files</u> property. The <u>OleDragDrop</u> event notifies your application that the user drags some data on the control. The ExDataObjectFiles object supports the following properties and methods:

Name	Description
Add	Adds a filename to the Files collection
<u>Clear</u>	Removes all file names in the collection.
<u>Count</u>	Returns the number of file names in the collection.
<u>ltem</u>	Returns an specific file name.
<u>Remove</u>	Removes an specific file name.

method ExDataObjectFiles.Add (FileName as String)

Adds a filename to the Files collection

Туре	Description
FileName as String	A string expression that indicates a filename.

Use Add method to add your files to ExDataObject object. The <u>OleStartDrag</u> event notifies your application that the user starts dragging items.

method ExDataObjectFiles.Clear ()

Removes all file names in the collection.

Туре

Description

Use the Clear method to remove all filenames from the collection.

property ExDataObjectFiles.Count as Long

Returns the number of file names in the collection.

Туре	Description
Long	A long value that indicates the count of elements into collection.

You can use "for each" statements if you are going to enumerate the elements into ExDataObjectFiles collection.

property ExDataObjectFiles.Item (Index as Long) as String

Returns a specific file name given its index.

Туре	Description
Index as Long	A long expression that indicates the filename's index.
String	A string value that indicates the filename.

method ExDataObjectFiles.Remove (Index as Long)

Removes a specific file name given its index into collection.

Туре	Description
Index as Long	A long expression that indicates the index of filename into collection.

Use <u>Clear</u> method to remove all filenames,.

ExPicture object

The ExPicture object identifies an icon or a picture to be displayed using the <u>Pictures</u> or <u>ExtraPictures</u> property of the Event object. The <u>ExPictures</u> collection is accessible through the <u>Pictures</u> property of the control. The ExPicture object supports the following properties and methods:

Name	Description
Content	Indicates the picture's content as it was previously added.
Enabled	Indicates whether the picture shows as enabled or disabled.
<u>Height</u>	Specifies the height of the picture, in pixels.
<u>lcon</u>	Specifies the index of the icon to be shown.
<u>Key</u>	Indicates the key of the picture.
<u>Picture</u>	Retrieves or sets a graphic to be displayed in the picture.
ShowHandCursor	Indicates whether the hand cursor is shown if the cursor hovers the picture.
Width	Specifies the width of the picture, in pixels.

property ExPicture.Content as Variant

Indicates the picture's content as it was previously added.

Туре	Description
Variant	A VARIANT expression that specifies the content of the picture, as being added by the <u>Add</u> method using the Picture parameter,

The Content property indicates the content of the picture, as being added by the <u>Add</u> method using the Picture parameter. This property is read-only, instead you can replace the picture's content using the Add method, with the same key or passing a different PictureDisp object to <u>Picture</u> property. The <u>Key</u> property indicates the key of the picture. The <u>Icon</u> property indicates the index of the icon in the <u>Images</u> collection to be used instead for displaying the picture object.

property ExPicture.Enabled as Boolean

Indicates whether the picture shows as enabled or disabled.

Туре	Description
Boolean	A Boolean expression that specifies whether the containing picture is displayed enabled or as disabled.

By default, the Enabled property is True, which makes the loaded picture to be shown as enabled. Use the Enabled property to show the containing picture as disabled. A disabled picture shows as grayed. The <u>ShowHandCursor</u> property indicates whether the hand cursor is shown when the cursor hovers a picture in the event's body.

property ExPicture.Height as Long

Specifies the height of the picture, in pixels.

Туре	Description
Long	A Long expression that specifies the height of the picture to be displayed, in pixels.

The Height property is initialized with the height of the loaded picture, once the <u>Add</u> method is called. You can use the Height/<u>Width</u> property to change the size of the picture when it is displayed on the event's body using the <u>Pictures</u> and <u>ExtraPictures</u> properties. If using the HTML tag to display the pictures/icons, the pic1:32, the 32 indicates that the size 32x32 should be used to display the image.

property ExPicture.lcon as Long

l

Specifies the index of the icon to be shown.

Туре	Description
Long	A Long expression that specifies the index of the icon to be displayed by the ExPicture object

By default, the Icon property is 0, which indicates no icon is displayed. If the Icon property is set to a value different than zero, it indicates the index of the icon to be displayed. The <u>Images</u> method of the control loads a collection of icons to be used in the control. No icon is displayed, if the index if not valid. Once the Icon property is set, the old content of the picture is unloaded. Setting the Icon property changes the <u>Width</u> and <u>Height</u> properties of the ExPicture object to 16.

property ExPicture.Key as String

Indicates the key of the picture.

Туре	Description
String	A String expression that indicates the key of the picture.

The Key property specifies the key of the picture to be displayed if included in the <u>Pictures</u> or <u>ExtraPictures</u> property of the Event object. The Key property is read-only. You need to remove and add a new ExPicture object, to replace the picture's key. The Key property is being identified with the Key parameter being used once the <u>Add</u> method is called.

property ExPicture.Picture as IPictureDisp

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

Туре	Description
IPictureDisp	A PictureDisp object that describes the image of the picture to be displayed.

You can use the Picture property to replace the picture's representation, without replacing the picture using the Add method. The <u>Content</u> property is read-only, instead you can replace the picture's content using the Add method, with the same key or passing a different PictureDisp object to Picture property. The <u>Content</u> property indicates the content of the picture, as being added by the Add method using the Picture parameter. The <u>Key</u> property indicates the key of the picture. The <u>Icon</u> property indicates the index of the icon in the <u>Images</u> collection to be used instead for displaying the picture object.

property ExPicture.ShowHandCursor as Boolean

Indicates whether the hand cursor is shown if the cursor hovers the picture.

Туре	Description
Boolean	A Boolean expression that specifies whether the control shows a hand cursor when the mouse is hovering the picture.

By default, the ShowHandCursor property is True. You can use the ShowHandCursor property to add clickable pictures to your event/appointment. The <u>PictureClick</u> event notifies your application once a picture is clicked in the event's body. The <u>PictureFromPoint</u> property indicates the key of the picture from the cursor. You can use the <u>EventFromPoint</u> property to get the <u>Event</u> object from the cursor. Also, the control displays the hand cursor when the mouse hovers an anchor element <a>.
property ExPicture.Width as Long

Specifies the width of the picture, in pixels.

Туре	Description
Long	A Long expression that specifies the width of the picture to be displayed, in pixels.

The Width property is initialized with the width of the loaded picture, once the <u>Add</u> method is called. You can use the <u>Height</u>/Width property to change the size of the picture when it is displayed on the event's body using the <u>Pictures</u> and <u>ExtraPictures</u> properties. If using the HTML tag to display the pictures/icons, the pic1:32, the 32 indicates that the size 32x32 should be used to display the image.

ExPictures object

The ExPictures collection holds a collection of icons, pictures that can be displayed using the <u>Pictures</u> and <u>ExtraPictures</u> properties of the Event object. The <u>Images</u> method loads icons to the control, <u>HTMLPicture</u> assigns a key to a picture object. The ExPictures collection can be accessed through the <u>Pictures</u> property of the control.

The control can display icons, pictures several ways as follows:

- Using the HTML tag, if used in any label properties such as LongLabel or ExtraLabel property. The ShortLabel property can NOT display images or HTML font attributes. For instance, this option can be used to display a default icon or picture for all events in the control using the DefaultEventLongLabel property. Also, the TAG can be used to any label or caption property that supports HTML format.
- Using the <u>Pictures</u> property of the Event object, which indicates a list of pictures from Pictures collection to be displayed on the event's body. The <u>PicturesAlign</u> property indicates the alignment of the pictures relative to the borders of the event. The hand cursor is shown, if the ExPicture.<u>ShowCursorHand</u> property is set, when the cursor hovers the picture. The <u>PictureClick</u> event is fired if the user clicks a picture on event's picture. You can use the <u>PictureFromPoint</u> property to get the identifier of the picture from the cursor.
- Using the <u>ExtraPictures</u> property of the Event object, which indicates a list of pictures from Pictures collection to be displayed on the event's body. The <u>ExtraPicturesAlign</u> property indicates the alignment of the pictures relative to the borders of the event. The hand cursor is shown, if the ExPicture.<u>ShowCursorHand</u> property is set, when the cursor hovers the picture. The <u>PictureClick</u> event is fired if the user clicks a picture on event's picture. You can use the <u>PictureFromPoint</u> property to get the identifier of the picture from the cursor.

The ExPictures collection supports the following methods and properties:

Name	Description
<u>Add</u>	Adds a Picture object to the collection and returns a reference to the newly created object.
<u>Clear</u>	Removes all objects in a collection.
<u>Count</u>	Returns the number of objects in a collection.
<u>ltem</u>	Returns a specific Picture of the Picture collection, giving its key.
<u>Remove</u>	Removes a specific member from the Pictures collection.

method ExPictures.Add (Key as String, Picture as Variant)

Adds a Picture object to the collection and returns a reference to the newly created object.

Туре	Description
Key as String	A String expression that indicates the key of the image to be added. <i>If Key and Picture parameters are both empty, the Pictures collection is cleared.</i>
Picture as Variant	 The Picture expression can be one of the followings: a long expression that specifies the index of the icon to be displayed instead. The Images method loads icons to the control a string expression that indicates the path to the picture file, being loaded, BMP, JPG, PNG, GIF and so on/ 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 no picture is added.
Return	Description
<u>ExPicture</u>	An ExPicture object being created, that holds the picture being loaded.

The <u>Images</u> method loads icons to the control, <u>HTMLPicture</u> assigns a key to a picture object, and the Pictures collection handles the identifiers of the pictures that can be used in the Pictures or <u>ExtraPictures</u> properties.

In order to display an icon or a picture in the control you need first to load the icons or the pictures you plan to display, using the <u>Images</u> method, <u>HTMLPicture</u>, or <u>Add</u> method of the ExPictures collection. The Images collection can display only 16x16 icons, while the <u>HTMLPicture</u>, or Add method can load and display custom sized pictures. The <u>Width/Height</u> property specifies the width and height of the picture to be displayed in the event's body.

The event can display icons, pictures several ways as follows:

- Using the HTML tag, if used in any label properties such as LongLabel or ExtraLabel property. The ShortLabel property can NOT display images or HTML font attributes. For instance, this option can be used to display a default icon or picture for all events in the control using the DefaultEventLongLabel property. Also, the TAG can be used to any label or caption property that supports HTML format.
- Using the <u>Pictures</u> property of the Event object, which indicates a list of pictures from Pictures collection to be displayed on the event's body. The <u>PicturesAlign</u> property indicates the alignment of the pictures relative to the borders of the event. The hand cursor is shown, if the ExPicture.<u>ShowCursorHand</u> property is set, when the cursor hovers the picture. The <u>PictureClick</u> event is fired if the user clicks a picture on event's picture. You can use the <u>PictureFromPoint</u> property to get the identifier of the picture from the cursor.
- Using the <u>ExtraPictures</u> property of the Event object, which indicates a list of pictures from Pictures collection to be displayed on the event's body. The <u>ExtraPicturesAlign</u> property indicates the alignment of the pictures relative to the borders of the event. The hand cursor is shown, if the ExPicture.<u>ShowCursorHand</u> property is set, when the cursor hovers the picture. The <u>PictureClick</u> event is fired if the user clicks a picture on event's picture. You can use the <u>PictureFromPoint</u> property to get the identifier of the picture from the cursor.

The <u>Pictures</u> and <u>ExtraPictures</u> may display one or more pictures at the time. The , character indicates the separator of pictures in the same line, while the *I* character divides the lines to show the pictures. For instance, "1,2" displays icon with the index 1 and 2 on the same line, while the "1/2,pic1" displays the first icon on the first line, the second icon and the picture pic1 on the second line.

The following samples displays a picture on the event's body:

VBA (MS Access, Excell...)

```
With Schedule1

.Calendar.Selection = #5/24/2012#

.Pictures.Add "pic1","c:\exontrol\images\zipdisk.gif"

.Events.Add(#5/24/2012 9:00:00 AM#,#5/24/2012 2:00:00 PM#).Pictures = "pic1"

End With
```

VB6

With Schedule1 .Calendar.Selection = #5/24/2012#

```
.Pictures.Add "pic1", "c:\exontrol\images\zipdisk.gif"
.Events.Add(#5/24/2012 9:00:00 AM#, #5/24/2012 2:00:00 PM#).Pictures = "pic1"
End With
```

VB.NET

```
With Exschedule1
.Calendar.Selection = #5/24/2012#
.Pictures.Add("pic1","c:\exontrol\images\zipdisk.gif")
.Events.Add(#5/24/2012 9:00:00 AM#,#5/24/2012 2:00:00 PM#).Pictures = "pic1"
End With
```

VB.NET for /COM

```
With AxSchedule1
.Calendar.Selection = #5/24/2012#
.Pictures.Add("pic1","c:\exontrol\images\zipdisk.gif")
.Events.Add(#5/24/2012 9:00:00 AM#,#5/24/2012 2:00:00 PM#).Pictures = "pic1"
End With
```

C++

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

#import <ExSchedule.dll>
using namespace EXSCHEDULELib;
*/
EXSCHEDULELib::ISchedulePtr spSchedule1 = GetDlgItem(IDC_SCHEDULE1)-
>GetControlUnknown();
spSchedule1->GetCalendar()->PutSelection("5/24/2012");
spSchedule1->GetPictures()->Add(L"pic1","c:\\exontrol\\images\\zipdisk.gif");
spSchedule1->GetEvents()->Add("5/24/2012 9:00:00 AM","5/24/2012 2:00:00 PM")-
>PutPictures(L"pic1");
```

```
Schedule1->Calendar->set_Selection(TVariant(TDateTime(2012,5,24).operator double()));
```

```
Schedule1->Pictures->Add(L"pic1",TVariant("c:\\exontrol\\images\\zipdisk.gif"));
Schedule1->Events->Add(TVariant(TDateTime(2012,5,24,9,00,00,0).operator
double()),TVariant(TDateTime(2012,5,24,14,00,00,0).operator double()))->Pictures =
L"pic1";
```

C#

```
exschedule1.Calendar.Selection =
Convert.ToDateTime("5/24/2012",System.Globalization.CultureInfo.GetCultureInfo("en-
US"));
exschedule1.Pictures.Add("pic1","c:\\exontrol\\images\\zipdisk.gif");
exschedule1.Events.Add(Convert.ToDateTime("5/24/2012 9:00:00
AM",System.Globalization.CultureInfo.GetCultureInfo("en-
US")),Convert.ToDateTime("5/24/2012 2:00:00
PM",System.Globalization.CultureInfo.GetCultureInfo("en-US"))).Pictures = "pic1";
```

JavaScript

```
<OBJECT classid="clsid:9B09E13D-7A88-4299-9DBE-383380435377"
id="Schedule1"></OBJECT>
<SCRIPT LANGUAGE="JScript">
Schedule1.Calendar.Selection = "5/24/2012";
Schedule1.Pictures.Add("pic1","c:\\exontrol\\images\\zipdisk.gif");
Schedule1.Events.Add("5/24/2012 9:00:00 AM","5/24/2012 2:00:00 PM").Pictures
= "pic1";
</SCRIPT>
```

C# for /COM

```
axSchedule1.Calendar.Selection =
Convert.ToDateTime("5/24/2012",System.Globalization.CultureInfo.GetCultureInfo("en-US"));
axSchedule1.Pictures.Add("pic1","c:\\exontrol\\images\\zipdisk.gif");
```

axSchedule1.Events.**Add**(Convert.ToDateTime("5/24/2012 9:00:00 AM",System.Globalization.CultureInfo.GetCultureInfo("en-US")),Convert.ToDateTime("5/24/2012 2:00:00 PM",System.Globalization.CultureInfo.GetCultureInfo("en-US"))).**Pictures** = "pic1";

X++ (Dynamics Ax 2009)

```
public void init()
{
    COM com_Event;
    anytype var_Event;
    ;
    super();

exschedule1.Calendar().Selection(COMVariant::createFromDate(str2Date("5/24/2012",2
    exschedule1.Pictures().Add("pic1","c:\\exontrol\\images\\zipdisk.gif");
    var_Event =
    COM::createFromObject(exschedule1.Events()).Add(COMVariant::createFromUtcDateTir
    9:00:00",213)),COMVariant::createFromUtcDateTime(str2Datetime("5/24/2012
    14:00:00",213))); com_Event = var_Event;
    com_Event.Pictures("pic1");
}
```

Delphi 8 (.NET only)

```
with AxSchedule1 do
begin
Calendar.Selection := '5/24/2012';
Pictures.Add('pic1','c:\exontrol\images\zipdisk.gif');
Events.Add('5/24/2012 9:00:00 AM','5/24/2012 2:00:00 PM').Pictures := 'pic1';
end
```

Delphi (standard)

```
with Schedule1 do
begin
Calendar.Selection := '5/24/2012';
Pictures.Add('pic1','c:\exontrol\images\zipdisk.gif');
Events.Add('5/24/2012 9:00:00 AM','5/24/2012 2:00:00 PM').Pictures := 'pic1';
end
```

VFP

```
with thisform.Schedule1
.Calendar.Selection = {^2012-5-24}
.Pictures.Add("pic1","c:\exontrol\images\zipdisk.gif")
.Events.Add({^2012-5-24 9:00:00},{^2012-5-24 14:00:00}).Pictures = "pic1"
endwith
```

dBASE Plus

```
local oSchedule,var_Event
oSchedule = form.Activex1.nativeObject
oSchedule.Calendar.Selection = "05/24/2012"
oSchedule.Pictures.Add("pic1","c:\exontrol\images\zipdisk.gif")
// oSchedule.Events.Add("05/24/2012 09:00:00","05/24/2012
14:00:00").Pictures = "pic1"
var_Event = oSchedule.Events.Add("05/24/2012 09:00:00","05/24/2012 14:00:00")
with (oSchedule)
TemplateDef = [Dim var_Event]
TemplateDef = var_Event
Template = [var_Event.Pictures = "pic1"]
endwith
```

XBasic (Alpha Five)

```
Dim oSchedule as P
Dim var_Event as P
```

```
oSchedule = topparent:CONTROL_ACTIVEX1.activex
```

```
oSchedule.Calendar.Selection = {05/24/2012}
oSchedule.Pictures.Add("pic1","c:\exontrol\images\zipdisk.gif")
' oSchedule.Events.Add({05/24/2012 09:00:00},{05/24/2012
14:00:00}).Pictures = "pic1"
var_Event = oSchedule.Events.Add({05/24/2012 09:00:00},{05/24/2012 14:00:00})
oSchedule.TemplateDef = "Dim var_Event"
oSchedule.TemplateDef = var_Event
oSchedule.Template = "var_Event.Pictures = \"pic1\""
```

Visual Objects

```
oDCOCX_Exontrol1:Calendar:Selection := SToD("20120524")
oDCOCX_Exontrol1:Pictures:Add("pic1","c:\exontrol\images\zipdisk.gif")
oDCOCX_Exontrol1:Events:Add(SToD("20120524 09:00:00"),SToD("20120524
14:00:00")):Pictures := "pic1"
```

PowerBuilder

```
OleObject oSchedule

oSchedule = ole_1.Object

oSchedule.Calendar.Selection = 2012-05-24

oSchedule.Pictures.Add("pic1","c:\exontrol\images\zipdisk.gif")

oSchedule.Events.Add(DateTime(2012-05-24,09:00:00),DateTime(2012-05-

24,14:00:00)).Pictures = "pic1"
```

method ExPictures.Clear ()

Removes all objects in a collection.

Туре

Description

The Clear method clears the ExPictures collection. The Clear method unloads the pictures being added through the Add method, but does not remove or clear any Pictures or ExtraPictures property of the Event object. The Clear method does not unload the icons being loaded using the Images method. The Remove method of the collection removes a specific picture from the collection. You can use the Add method, to replace an exiting picture, when using the same key. The ShowEventPictures property indicates whether the control displays the event's pictures associated using the Pictures or ExtraPictures property.

property ExPictures.Count as Long

Returns the number of objects in a collection.

Туре	Description
Long	A long expression that specifies the number of ExPicture object in the ExPictures collection.

The Count property counts the number of <u>ExPicture</u> object in the ExPictures collection. The <u>Clear</u> method clears or remove all elements in the ExPictures collection, so the Count will be 0.

property ExPictures.Item (Key as Variant) as ExPicture

Returns a specific Picture of the Picture collection, giving its key.

Туре	Description
Key as Variant	A string expression that specifies the key of the picture being accessed.
<u>ExPicture</u>	An ExPicture object being requested.

The Item property can be used to access the ExPicture object using its key. For instance, you can use the <u>PictureFromPoint</u> property to get the identifier of the picture from the cursor. Also, the <u>PictureClick</u> event notifies once a picture is being clicked. The Key parameter of the <u>PictureClick</u> event indicates the key of the picture being clicked.

The event can display icons, pictures several ways as follows:

- Using the HTML tag, if used in any label properties such as LongLabel or ExtraLabel property. The ShortLabel property can not display images or HTML font attributes. If the tag is included in a <a> HTML tag, you have a clickable image through the AnchorClick event. This option can be used to display a default icon or picture for all events in the control using the DefaultEventLongLabel property.
- Using the <u>Pictures</u> property of the Event object, which indicates a list of pictures from Pictures collection to be displayed on the event's body. The <u>PicturesAlign</u> property indicates the alignment of the pictures relative to the borders of the event. The hand cursor is shown, if the ExPicture.<u>ShowCursorHand</u> property is set, when the cursor hovers the picture. The <u>PictureClick</u> event is fired if the user clicks a picture on event's picture. You can use the <u>PictureFromPoint</u> property to get the identifier of the picture from the cursor.
- Using the <u>ExtraPictures</u> property of the Event object, which indicates a list of pictures from Pictures collection to be displayed on the event's body. The <u>ExtraPicturesAlign</u> property indicates the alignment of the pictures relative to the borders of the event. The hand cursor is shown, if the ExPicture.<u>ShowCursorHand</u> property is set, when the cursor hovers the picture. The <u>PictureClick</u> event is fired if the user clicks a picture on event's picture. You can use the <u>PictureFromPoint</u> property to get the identifier of the picture from the cursor.

The <u>Picture</u> and <u>ExtraPictures</u> may display one or more pictures at the time. The , character indicates the separator of pictures in the same line, while the / character divides the lines to show the pictures. For instance, "1,2" displays icon with the index 1 and 2 on the same line, while the "1/2,pic1" displays the first icon on the first line, the second icon and the picture pic1 on the second line.

method ExPictures.Remove (Key as Variant)

Removes a specific member from the Pictures collection.

Туре	Description
Key as Variant	A string expression that specifies the key of the picture to be removed.

The Remove method of the collection removes a specific picture from the collection. The <u>Clear</u> method clears the ExPictures collection. The Remove/<u>Clear</u> method unloads the pictures being added through the <u>Add</u> method, but does not remove or clear any <u>Pictures</u> or <u>ExtraPictures</u> property of the Event object. The Clear method does not unload the icons being loaded using the <u>Images</u> method. You can use the <u>Add</u> method, to replace an exiting picture, when using the same key. The <u>ShowEventPictures</u> property indicates whether the control displays the event's pictures associated using the <u>Pictures</u> or <u>ExtraPictures</u> property.

Group object

A Group object holds information about the event's group. The control can display events on different groups aligned on columns. The <u>Add</u> method of the <u>Groups</u> collection adds a new group to the control. The <u>GroupID</u> property of the Event specifies the identifier of the Group that hosts the event. The Groups collection is accessible through the <u>Groups</u> property of the control.

The Group object supports the following properties and methods.

Name	Description
<u>Alignment</u>	Indicates the alignment of the caption/title of the Group object.
<u>CalendarHighlightEvent</u>	Gives access to the Highlight object, so you can customize highlighting the events of this group, in the calendar panel.
Caption	Specifies the HTML caption of the group.
EventBackColor	Specifies the background color or the visual appearance of the events in the same group.
<u>EventForeColor</u>	Specifies the foreground color of the events in the same group.
<u>EventPattern</u>	Specifies the pattern to display the events in the same group.
HeaderBackColor	Specifies the background color or the visual appearance of the header's group.
HeaderForeColor	Specifies the foreground color of the header's group.
<u>HeaderPattern</u>	Specifies the pattern to show the group's header
ID	Gets or sets the identifier of the current group.
Index	Indicates the index of the Group object in the Groups collection.
Position	Gets or sets the position of the current group.
<u>ScheduleHighlightEvent</u>	Gives access to the Highlight object, so you can customize highlighting the events of this group, in the schedule panel.
Title	Indicates the title of the group.
<u>ToolTip</u>	Indicates the tooltip of the group.
<u>UserData</u>	Indicates any extra data associated with the Group object.
Visible	Indicates whether the group is visible or hidden.
<u>Width</u>	Gets or sets the width of the current group.

property Group.Alignment as ContentAlignmentEnum

Indicates the alignment of the caption/title of the Group object.

Туре

Description

<u>ContentAlignmentEnum</u> A ContentAlignmentEnum expression that specifies the alignment of the caption/title of the Group.

By default, the Alignment property is exTopCenter. The Alignment property is applied to group's client area not to the group's header only. In other words, if you set the Alignment to exBottomCenter, the group's title/caption is being displayed on the bottom of the group, instead of the group's header. The <u>HeaderGroupHeight</u> property indicates the height of the group's header relative to the control's font height. The group's header does not clip its caption or title to the header section, to allow displaying the HTML text or pictures on the Group's client as in the following screen shot. The <u>Caption</u> property indicates the HTML caption to be displayed on the Group's header. The <u>Title</u> property indicates the title to be shown when the user drop down the grouping button. The control displays Caption or Title on the control's header based on the date's size. If the date is too small, the title may be displayed instead caption. By default, You can resize the schedule view, by dragging the mouse while clicking the middle mouse button.



property Group.CalendarHighlightEvent as Highlight

Gives access to the Highlight object, so you can customize highlighting the events of this group, in the calendar panel.

Туре	Description
<u>Highlight</u>	A Highlight object to customize the dates with events.

The CalendarHighlightEvent property indicates the visual aspect of the dates with events in the same group, in the calendar panel. The CalendarHighlightEvent property has effect only, the the <u>GroupHighlightEvent</u> property is True. The <u>GroupHighlightEvent</u> property specifies if events are highlighted using the <u>HighlightEvent</u> property (False), or using the <u>CalendarHighlightEvent</u> property of the <u>Group</u> that event belongs to. The <u>GroupID</u> property indicates the identifier of the event's group. The <u>ScheduleHighlightEvent</u> property specifies the visual appearance of dates with events in the schedule panel.

Using the CalendarHighlightEvent object a date with events can combine one or more of the following options:

- **bold**, **<u>Bold</u> property renders as bold text**
- *italic*, <u>Italic</u> property renders as italic text
- <u>underline</u>, <u>Underline</u> property underlines the text
- strikeout, StrikeOut property shows the text with a horizontal line through its center
- change the font SiZe, <u>FontSize</u> property indicates the size of the font to display the text
- change the **font**, using the Font property
- change the text's foreground color, using the ForeColor property
- change the text's background color, using the BackColor property
- shows a pattern using the Pattern property

The following screen shot shows the dates with events when <u>GroupHighlightEvent</u> property is True:

									Thursday,Ma	ay 17,2012 📃	Fri	day,May 1	8,2012	1
									7 Prima	1 Pro	7 1	Prima	1 Pro	1
•		1	May :	2012	2		•	08:00 AM :30						
	Мо	Tu	We	Th	Fr	Sa	Su	09:00 AM	9:00 AM -		9:15 A	M -		
18		1	2	3	4	5	6	10:00 AM	11.15 AM	10:00 AM -	12:00	PM		
19	7	8	9	10	11	12	13	:30		11:30 AM				
20	14	15	16	17	18	19	20	11:00 AM :30						
21	21	22	23	24	25	26	27	12:00 PM		12:00 PM -				
22	28	29	30	31				:30 01:00 PM	12:15 PM - 2:00 PM	2:00 PM	12 PM -			
20								02:00 PM :30 03:00 PM			PM	1:45 PM - 3:45 PM		
			Tor	av)			:30 03:00 PM :30				3:45 PM		TILLING TO THE T

property Group.Caption as String

Specifies the HTML caption of the group.

Туре	Description
String	A String expression that specifies the HTML caption to be displayed on the group's header.

By default, the Caption property is "". The Caption parameter of the Add method indicates the caption to be shown on the group's header. At adding time, the <u>Title</u> property is initialized with the Caption property with no HTML tags. You can include icons or pictures in the group's Caption using the HTML tags. The <u>HeaderGroupHeight</u> property indicates the height of the group's header relative to the control's font height. The <u>Title</u> property indicates the title to be shown when the user drop down the grouping button. The control displays Caption or Title on the control's header based on the date's size. If the date is too small, the title may be displayed instead caption. By default, You can resize the schedule view, by dragging the mouse while clicking the middle mouse button. The <u>ShortLabel</u>, <u>LongLabel</u> or <u>ExtraLabel</u> property of the Event may display automatically the group's Caption if the <%=%262%> tag is included in the label.



The Caption property supports the following HTML tags:

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

```
The control supports expandable HTML captions feature which allows you to expand(show)/collapse(hide) different information using <a ;exp=> or <a ;e64=>
```

anchor tags. The exp/e64 field of the anchor stores the HTML line/lines to show once the user clicks/collapses/expands the caption.

- exp, stores the plain text to be shown once the user clicks the anchor, such as "<a ;exp=show lines>"
- e64, encodes in BASE64 the HTML text to be shown once the user clicks the anchor, such as "<a/li>

;e64=gA8ABmABnABjABvABshIAOQAEAAHAAGESikWio+ABzABohp3iELABpABu " that displays show lines- in gray when the user clicks the + anchor. The "gA8ABmABnABjABvABshIAOQAEAAHAAGESikWio+ABzABohp3iELABpABuABljY string encodes the "<fgcolor 808080>show lines<a>-</fgcolor>" The Decode64Text/Encode64Text methods of the eXPrint can be used to decode/encode e64 fields.

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

- ... displays portions of text with a different font and/or different size. For instance, the "bit" draws the bit text using the Tahoma font, on size 12 pt. If the name of the font is missing, and instead size is present, the current font is used with a different size. For instance, "bit" displays the bit text using the current font, but with a different size.
- <fgcolor rrggbb> ... </fgcolor> or <fgcolor=rrggbb> ... </fgcolor> displays text with a specified foreground color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- <bgcolor rrggbb> ... </bgcolor> or <bgcolor=rrggbb> ... </bgcolor> displays text with a specified background color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- <solidline rrggbb> ... </solidline> or <solidline=rrggbb> ... </solidline> draws a solidline 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
- number[:width] 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.
- key[:width] 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 FFFFF;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:

outline anti-aliasing

property Group.EventBackColor as Color

Specifies the background color or the visual appearance of the events in the same group.

Туре	Description
Color	A Color expression that specifies the background color to show the event's body. The last 7 bits in the high significant byte of the color to indicates the identifier of the skin being used. Use the <u>Add</u> 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 EventBackColor property specifies the event's background color if it belongs to a group. The <u>ApplyGroupingColors</u> property indicates whether the <u>EventForeColor</u>, EventBackColor and <u>EventPattern</u> properties of the Group object are being applied to events. The <u>BodyBackColor</u> property specifies the background color of the event's body. The <u>BodyEventBackColor</u> property specifies the background color to show the body for all events. The <u>GroupID</u> property of the Event specifies the identifier of the Group that owns the event. The <u>HeaderBackColor</u> property can be used to specify a different visual appearance of the group's header.

The control displays groups if

- <u>ShowGroupingEvents</u> property is True
- The <u>Groups</u> collection has elements. By default, the Groups collection contains no Group objects.

property Group.EventForeColor as Color

Specifies the foreground color of the events in the same group.

Туре	Description
Color	A Color expression that specifies the event's foreground color.

By default, the EventForeColor property is 0. The EventForeColor property has effect only if it is not-zero. The EventForeColor property specifies the event's foreground color if it belongs to a group. The BodyForeColor property specifies the foreground color to show the labels on the event. The ApplyGroupingColors property indicates whether the EventForeColor, EventBackColor and EventPattern, properties of the Group object are being applied to events. The HeaderForeColor property can be used to specify a different foreground color on the group's header.

The control displays groups if

- ShowGroupingEvents property is True
- The <u>Groups</u> collection has elements. By default, the Groups collection contains no Group objects.

property Group.EventPattern as Pattern

Specifies the pattern to display the events in the same group.

Туре	Description
Pattern	A Pattern object that can be accessed to specify a different pattern to be shown on events of the group.

The EventPattern property specifies the event's pattern color if it belongs to a group. The <u>ApplyGroupingColors</u> property indicates whether the <u>EventForeColor</u>, <u>EventBackColor</u> and EventPattern, properties of the Group object are being applied to events. The <u>BodyPattern</u> property specifies the pattern of the event's body. The <u>GroupID</u> property of the Event specifies the identifier of the Group that owns the event. The <u>HeaderPattern</u> property can be used to specify a different pattern on the group's header.

The control displays groups if

- ShowGroupingEvents property is True
- The <u>Groups</u> collection has elements. By default, the Groups collection contains no Group objects.

property Group.HeaderBackColor as Color

Specifies the background color or the visual appearance of the header's group.

Туре	Description
Color	A Color expression that specifies the background color to show on the group's header. The last 7 bits in the high significant byte of the color to indicates the identifier of the skin being used. Use the <u>Add</u> 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 HeaderBackColor property specifies the background color of the group's header. The <u>HeaderForeColor</u> property specifies the group's header foreground color. The <u>EventBackColor</u> property specifies the colors to be applied on the events of the group. The <u>HeaderGroupHeight</u> property indicates the height of the group's header relative to the control's font height. The group's header does not clip its caption or title to the header section, to allow displaying the HTML text or pictures on the Group's client as in the following screen shot. The <u>Caption</u> property indicates the title to be shown when the user drop down the grouping button. The control displays Caption or Title on the control's header based on the date's size. If the date is too small, the title may be displayed instead caption. By default, You can resize the schedule view, by dragging the mouse while clicking the middle mouse button.

property Group.HeaderForeColor as Color

Specifies the foreground color of the header's group.

Туре	Description
Color	A Color expression that specifies the foreground color to show the caption or the group's title.

The HeaderForeColor property specifies the background color of the group's header. The HeaderBackColor property specifies the group's header background color. The EventForeColor property specifies the colors to be applied on the events of the group. The HeaderGroupHeight property indicates the height of the group's header relative to the control's font height. The group's header does not clip its caption or title to the header section, to allow displaying the HTML text or pictures on the Group's client as in the following screen shot. The Caption property indicates the title to be shown when the user drop down the grouping button. The control displays Caption or Title on the control's header based on the date's size. If the date is too small, the title may be displayed instead caption. By default, You can resize the schedule view, by dragging the mouse while clicking the middle mouse button.

property Group.HeaderPattern as Pattern

Specifies the pattern to show the group's header

Туре	Description
Pattern	A Pattern object to specify the pattern to be shown on the group's header.

By default, the <u>Type</u> property of the HeaderPattern property is exPatternEmpty which indicates that no pattern is shown. The <u>Type</u> property indicates the pattern to display on the group's header. The <u>Color</u> property indicates the color to display the pattern. The <u>FrameColor</u> property indicates the color to show the frame, if the exPatternFrame flag is included in the Type property. The <u>EventPattern</u> property gives access to the group's Header pattern. The <u>HeaderGroupHeight</u> property indicates the height of the group's header relative to the control's font height. The group's header does not clip its caption or title to the header section, to allow displaying the HTML text or pictures on the Group's client as in the following screen shot. The <u>Caption</u> property indicates the title to be shown when the user drop down the grouping button. The control displays Caption or Title on the control's header based on the date's size. If the date is too small, the title may be displayed instead caption. By default, You can resize the schedule view, by dragging the mouse while clicking the middle mouse button.

property Group.ID as Long

Gets or sets the identifier of the current group.

Туре	Description
Long	A Long expression that specifies the identifier of the Group object,

The ID property indicates the identifier of the Group. A Group object may hosts multiple Event objects, when the <u>GroupID</u> for each event is the same as the ID property. The ID parameter of the <u>Add</u> method indicates the ID of the Group being added. You can use the <u>Item</u> property of the Groups collection to access a group giving its identifier. Use the <u>GroupID</u> property of the <u>NonworkingTime</u> object to apply the non-working time to the specified groups.

The control displays groups if:

- ShowGroupingEvents property is True
- The <u>Groups</u> collection has elements. By default, the Groups collection contains no Group objects.

If the control displays groups the GroupID property of the newly created event is automatically updated with the group where the event has been created. The <u>AllowMoveEventToOtherGroup</u> property specifies whether the user can move an event from a group to another at runtime. Use the <u>Add</u> method of the Groups collection to add new groups to the control.

property Group.Index as Long

Indicates the index of the Group object in the Groups collection.

Туре	Description
Long	A Long expression that specifies the index of the Group object in the Groups collection.

The Index property indicates the index of the Group object in the Groups collection. You can use the <u>ItemByIndex</u> property of the Groups collection to access a group giving its index. The Index and <u>ID</u> properties are different. The Index property goes from 0 to <u>Count</u> - 1, while ID property is chosen by the user. The Index property is a read only property.

property Group.Position as Long

Gets or sets the position of the current group.

Туре	Description
Long	A Long expression that specifies the position of the Group when displaying in the control. The position starts at 0, as being the first visible, group, 1, the second visible group and so on.

The Position property can be used to programmatically change the Group's position by code. You can enumerate the Group as being displayed using the <u>ItemByPos</u> property of Groups collection. By default, the user can change the Group's position by dragging the Group's Header to a new position. The <u>AllowMoveGroup</u> property specifies the combination of keys so user can move the Groups at runtime. The <u>Visible</u> property of the Group specifies whether the Group is visible in the schedule view, and un-checked, in the drop down grouping panel. The <u>AllowResizeGroup</u> property specifies whether the user can resize a group at runtime.

property Group.ScheduleHighlightEvent as Highlight

Gives access to the Highlight object, so you can customize highlighting the events of this group, in the schedule panel.

TypeDescriptionHighlightA Highlight object to customize the dates with events.

The ScheduleHighlightEvent property specifies the visual appearance of dates with events in the schedule panel. The <u>HighlightEvent</u> property and <u>CalendarHighlightEvent</u> property highlights the dates in the calendar panel only. The ScheduleHighlightEvent property has effect only when the schedule view is resized to the minimum (no date header is shown, no time scale is shown, the entire year is visible).

Using the CalendarHighlightEvent object a date with events can combine one or more of the following options:

- **bold**, **Bold** property renders as bold text
- *italic*, <u>Italic</u> property renders as italic text
- <u>underline</u>, <u>Underline</u> property underlines the text
- strikeout, StrikeOut property shows the text with a horizontal line through its center
- change the font SIZE, <u>FontSize</u> property indicates the size of the font to display the text
- change the **font**, using the <u>Font</u> property
- change the text's <u>foreground</u> color, using the <u>ForeColor</u> property
- change the text's background color, using the BackColor property
- shows a pattern using the Pattern property

The following screen shot shows the dates with events of the same group (gray only):

-	15	14	13	12	11	10	9
	22	21	20	19	18	17	16
	29	28	27	26	25	24	23
				20			April 30
	6	5	4	3	2	May 1	
	13	12	11	10	9	8	7
	20	19	18	17	16	15	14
=	27	26	25	24	23	22	21
			14 1 mm	May 31	30	29	28
	3	2	June 1				
	10	9	8	7	6	5	4
	17	16	15	14	13	12	11
	24	23	22	21	20	19	18
		June 30	29	28	27	26	25
	July 1		24 	2. 			1
	8	7	6	5	4	3	2
-	15	14	13	12	11	10	9

The following screen shot shows the dates with events of the same group (red only):

	15	14	13	12	11	10	9
	22	21	20	19	18	17	16
	29	28	27	26	25	24	23
							April 30
	6	5	4	3	2	May 1	
	13	12	11	10	9	8	7
_	20	19	18	17	16	15	14
=	27	26	25	24	23	22	21
			S	May 31	30	29	28
	3	2	June 1				
	10	9	8	7	6	5	4
	17	16	15	14	13	12	11
	24	23	22	21	20	19	18
_		June 30	29	28	27	26	25
	July 1			26 			
	8	7	6	5	4	3	2
-	15	14	13	12	11	10	9

The following screen shot shows the dates with events of different groups (combined):

	15	14	13	12	11	10	9
	22	21	20	19	18	17	16
	29	28	27	26	25	24	23
		80					April 30
	6	5	4	3	2	May 1	
	13	12	11	10	9	8	7
	20	19	18	17	16	15	14
=	27	26	25	24	23	22	21
				May 31	30	29	28
	3	2	June 1	1.00			
	10	9	8	7	6	5	4
	17	16	15	14	13	12	11
	24	23	22	21	20	19	18
_		June 30	29	28	27	26	25
7	July 1			15			
	8	7	6	5	4	3	2
-	15	14	13	12	11	10	9

property Group.Title as String

Indicates the title of the group.

Туре	Description
String	A String expression that specifies the HTML title to be displayed on the drop down panel, when user clicks the grouping button.

By default, the Title property is initialized with the <u>Caption</u> parameter of the <u>Add</u> method, with no HTML tags. The Title property indicates the title to be shown when the user drop down the grouping button. The control displays <u>Caption</u> or Title on the control's header based on the date's size. If the date is too small, the title may be displayed instead caption. By default, You can resize the schedule view, by dragging the mouse while clicking the middle mouse button. The <u>ShortLabel</u>, <u>LongLabel</u> or <u>ExtraLabel</u> property of the Event may display automatically the group's Title if the <%=%263%> tag is included in the label. The <u>DisplayGroupingButton</u> property indicates whether the header of the date displays the grouping button. The list of available groups is displayed on a drop down panel, once the user clicks the grouping/filtering button. The drop down list shows the Title for each group found.

property Group.ToolTip as String

Indicates the tooltip of the group.

Туре	Description
String	A String expression that specifies the HTML tooltip to be shown when the cursor hovers the Group's header.

By default, the ToolTip property is "", which indicates no tooltip is being shown if the cursor hovers the group's header. The ToolTip property indicates the tooltip to be shown when the cursor hovers the event. The <u>ShowToolTip</u> method can be used during the <u>MouseMove</u> event to display a custom tooltip. Use the <u>ToolTipPopDelay</u> property specifies the period in ms of time the ToolTip remains visible if the mouse pointer is stationary within a control. Use the <u>ToolTipFont</u> property to change the tooltip's font.

The property supports the following built-in HTML tags:

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

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

- exp, stores the plain text to be shown once the user clicks the anchor, such as "<a ;exp=show lines>"
- $\circ~$ e64, encodes in BASE64 the HTML text to be shown once the user clicks the anchor, such as "<a $\!$

;e64=gA8ABmABnABjABvABshIAOQAEAAHAAGESikWio+ABzABohp3iELABpABu " that displays show lines- in gray when the user clicks the + anchor. The "gA8ABmABnABjABvABshIAOQAEAAHAAGESikWio+ABzABohp3iELABpABuABljY string encodes the "<fgcolor 808080>show lines<a>-</fgcolor>" The Decode64Text/Encode64Text methods of the eXPrint can be used to decode/encode e64 fields. Any ex-HTML caption can be transformed to an expandable-caption, by inserting the anchor ex-HTML tag. For instance, "<solidline>Header</solidline>
Line1<r><a ;exp=show lines>+
Line2
Line3" shows the Header in underlined and bold on the first line and Line1, Line2, Line3 on the rest. The "show lines" is shown instead of Line1, Line2, Line3 once the user clicks the + sign.

- ... displays portions of text with a different font and/or different size. For instance, the "bit" draws the bit text using the Tahoma font, on size 12 pt. If the name of the font is missing, and instead size is present, the current font is used with a different size. For instance, "bit" displays the bit text using the current font, but with a different size.
- **<fgcolor rrggbb>** ... **</fgcolor>** or **<**fgcolor=rrggbb> ... **</**fgcolor> displays text with a specified foreground color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- <bgcolor rrggbb> ... </bgcolor> or <bgcolor=rrggbb> ... </bgcolor> displays text
 with a specified background color. The rr/gg/bb represents the red/green/blue values of
 the color in hexa values.
- <solidline rrggbb> ... </solidline> or <solidline=rrggbb> ... </solidline> draws a solidline 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
- number[:width] 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.
- key[:width] 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 FFFFF;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 Group.UserData as Variant

Indicates any extra data associated with the Group object.

Туре	Description
Variant	A VARIANT expression that specifies any extra data to be associated with the group.

By default, the UserData property is empty or nothing. You can use the UserData property to assign any extra data to your Group object.

property Group.Visible as Boolean

Indicates whether the group is visible or hidden.

Туре	Description
Boolean	A Boolean expression that specifies whether the Group is visible or hidden in the schedule view.

By default, the Visible property is False. The Visible property of the Group specifies whether the Group is visible in the schedule view, and un-checked, in the drop down grouping panel. The <u>AllowMoveGroup</u> property specifies the combination of keys so user can move the Groups at runtime. The <u>AllowResizeGroup</u> property specifies whether the user can resize a group at runtime. The <u>EventBackColor</u>, <u>EventForeColor</u> or <u>EventPattern</u> property specifies the colors to be applied on the events of the group. The <u>ApplyGroupingColors</u> property indicates whether the EventForeColor, EventBackColor and EventPattern properties of the Group object are being applied to events. You can use the <u>Width</u> property to change the width of the Group by code.

The control displays groups if

- <u>ShowGroupingEvents</u> property is True
- The <u>Groups</u> collection has visible elements. By default, the Groups collection contains no Group objects (Visible on True)

property Group.Width as Long

Gets or sets the width of the current group.

Туре	Description
Long	A Long expression that specifies the width of the group, in pixels.

By default, the schedule view splits the groups in equal parts. You can use the Width property to change the width of the Group by code. The <u>Visible</u> property specifies whether a group is shown or hidden. The Width property has no effect if the Group is not visible. The <u>AllowResizeGroup</u> property indicates whether the user can change the width of the group at runtime,

Groups object

The Groups collection holds a collection of Group objects. The control can displays events on different columns, each column being identified as a Group. The Groups collection is accessible through the <u>Groups</u> property.

The following sample shows the grouping header and the grouping panel:



The Groups collection supports the following properties and methods:

Name	Description
Add	Adds a Group object to the collection and returns a reference to the newly created object.
<u>Clear</u>	Removes all objects in a collection.
Count	Returns the number of objects in a collection.
<u>ltem</u>	Returns a specific Group of the Groups collection, giving its identifier.
<u>ItemByIndex</u>	Returns a specific Group of the Groups collection, giving its index.
<u>ItemByPos</u>	Returns a specific Group of the Groups collection, giving its position.

method Groups.Add (ID as Long, Caption as String)

Adds a Group object to the collection and returns a reference to the newly created object.

Туре	Description
ID as Long	A Long expression that specifies the identifier of the Group. In order to assign an event to the same group you need to use the same identifier in the <u>GroupID</u> property of the Event.
Caption as String	A String expression that indicates the HTML format to be shown on the Group's header.
Return	Description
<u>Group</u>	A Group object being created.

Use the Add method of the Groups collection to add new groups to the control. Use the <u>Visible</u> property to make the group visible in the schedule panel. The <u>DisplayGroupingButton</u> property indicates whether the header of the date displays the grouping button. The list of available groups is displayed on a drop down panel, once the user clicks the grouping/filtering button. The drop down list shows the <u>Title</u> for each group found. The ShowGroupingEvents property indicates whether the control displays events grouped by its <u>GroupID</u> property. The <u>ApplyGroupingColors</u> property specifies whether the control uses the Group's <u>EventBackColor</u> / <u>EventForeColor</u> / <u>EventPattern</u> properties to show the events in the groups. The <u>SingleGroupingView</u> property specifies whether the drop down panel shows radio buttons, instead check boxes, so the user can see all groups or only one group at the time. The <u>GroupID</u> property specifies the identifier of the group where the event belongs. If the control displays groups the GroupID property of the newly created event is automatically updated with the group where the event has been created. The <u>AllowMoveEventToOtherGroup</u> property specifies whether the user can move an event from a group to another at runtime.

The control displays groups if:

- <u>ShowGroupingEvents</u> property is True
- The <u>Groups</u> collection has elements. By default, the Groups collection contains no Group objects.
- The Groups collection contains visible element, or Group object with the <u>Visible</u> property on True.

The following <u>Background</u> properties change the visual appearance of the drop down grouping panel:

• Background(exGroupingBackColor) / Background(exGroupingForeColor) changes the background and the foreground color of the panel.

- Background(exGroupingSelBackColor) / Background(exGroupingSelForeColor) changes the background and the foreground color of the selection in the panel.
- Background(exCheckBoxState0), Background(exCheckBoxState1), Background(exCheckBoxState2) changes the visual appearance for the control's check boxes.
- Background(exRadioButtonState0), Background(exRadioButtonState1), changes the visual appearance for the control's radio buttons.

The <u>Description</u>(exGroupBarAll) property changes the "(All)" predefined string, being displayed on the top of the drop down grouping/filtering panel.

method Groups.Clear ()

Removes all objects in a collection.

Туре

Description

The Clear method removes all Group objects in the Groups collection. The <u>Remove</u> method removes a specific Group object. The <u>ApplyGroupingColors</u> property specifies whether the control uses the Group's <u>EventBackColor</u> / <u>EventForeColor</u> / <u>EventPattern</u> properties to show the events in the groups. The <u>AllowMoveEventToOtherGroup</u> property specifies whether the user can move an event from a group to another at runtime. Use the <u>Visible</u> property to show or hide a group in the schedule panel.

property Groups.Count as Long

Returns the number of objects in a collection.

Туре	Description
Long	A Long expression that specifies the number of Group objects in the Groups collection.

The Count property counts the number of Group objects in the Groups collection. The <u>ItemByIndex</u>/Count properties can be used to enumerate the Groups as they has been added. The <u>ItemByPos</u>/Count properties can be used to enumerate the Groups as they are displayed. The <u>Visible</u> property specifies whether a Group is visible or hidden. The ItemByPos property gives the Group by position, no matter if it is visible or hidden. The <u>Item</u> property can be used to access a Group object giving its identifier. Use the <u>Add</u> method of the Groups collection to add new groups to the control. The Groups collection supports **for each statement**, so you can enumerate the groups using a code like *for each g in Groups*.

property Groups.Item (ID as Variant) as Group

Returns a specific Group of the Groups collection, giving its identifier.

Туре	Description
ID as Variant	A long expression that specifies the identifier of the group to be accessed.
Group	The Group object based on its identifier.

The Item property can be used to access a Group object giving its identifier. Use the <u>Add</u> method of the Groups collection to add new groups to the control. The <u>ItemByPos/Count</u> properties can be used to enumerate the Groups as they are displayed. The <u>Visible</u> property specifies whether a Group is visible or hidden. The ItemByPos property gives the Group by position, no matter if it is visible or hidden. The <u>ItemByIndex/Count</u> properties can be used to enumerate the Groups as they has been added. The <u>Count</u> property counts the number of Group objects in the Groups collection.

property Groups.ItemByIndex (Index as Long) as Group

Returns a specific Group of the Groups collection, giving its index.

Туре	Description
Index as Long	A long expression that specifies the index of the Group being requested
<u>Group</u>	A Group object being requested

The ItemByIndex/<u>Count</u> properties can be used to enumerate the Groups as they has been added. The <u>Count</u> property counts the number of Group objects in the Groups collection. The <u>ItemByPos/Count</u> properties can be used to enumerate the Groups as they are displayed. The <u>Visible</u> property specifies whether a Group is visible or hidden. The ItemByPos property gives the Group by position, no matter if it is visible or hidden. The <u>Item</u> property can be used to access a Group object giving its identifier. Use the <u>Add</u> method of the Groups collection to add new groups to the control.

property Groups.ItemByPos (Position as Long) as Group

Returns a specific Group of the Groups collection, giving its position.

Туре	Description
Position as Long	A long expression that specifies the position of the Group being requested
<u>Group</u>	The Group object being requested

The ItemByPos/Count properties can be used to enumerate the Groups as they are displayed. The <u>Visible</u> property specifies whether a Group is visible or hidden. The ItemByPos property gives the Group by position, no matter if it is visible or hidden. The <u>ItemByIndex/Count</u> properties can be used to enumerate the Groups as they has been added. The <u>Count</u> property counts the number of Group objects in the Groups collection. The <u>Item</u> property can be used to access a Group object giving its identifier. Use the <u>Add</u> method of the Groups collection to add new groups to the control.

method Groups.Remove (ID as Variant)

l

Removes a specific member from the Groups collection.

Туре	Description
ID as Variant	A Long expression that specifies the identifier of the Group to be removed.

The Remove method removes a specific Group object. Use the <u>Visible</u> property to show or hide a group in the schedule panel. The <u>Clear</u> method removes all Group objects in the Groups collection. The <u>ApplyGroupingColors</u> property specifies whether the control uses the Group's <u>EventBackColor</u> / <u>EventForeColor</u> / <u>EventPattern</u> properties to show the events in the groups. The <u>AllowMoveEventToOtherGroup</u> property specifies whether the user can move an event from a group to another at runtime.

Highlight object

The following properties uses a Highlight object:

- <u>HighlightEvent</u> property customizes visual appearance of the events, in the calendar panel
- <u>CalendarHighlightEvent</u> property customizes visual appearance of the events in the same group, in the calendar panel
- <u>ScheduleHighlightEvent</u> property specifies the visual appearance of dates with events in the schedule panel.

The Highlight object supports the following properties:

Name	Description
BackColor	Specifies the element's background color.
Bold	Renders as bold text.
<u>Font</u>	Retrieves or sets the text's font.
<u>FontSize</u>	Indicates the size of the font to display the text.
<u>ForeColor</u>	Specifies the element's foreground color.
<u>Italic</u>	Renders as italic text.
<u>Pattern</u>	Gives access to the element's Pattern object.
<u>StrikeOut</u>	Specifies that the text should appear as strikeout.
<u>Underline</u>	Underlines the text.

property Highlight.BackColor as Color

Specifies the element's background color.

Туре	Description
Color	A color expression that indicates the element'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.

By default, the BackColor property is 0, which means no background color is applied to the object. The BackColor property has effect only if a non-zero value is used. The ForeColor property specifies the element's foreground color.



property Highlight.Bold as Boolean

Renders as bold text.

l

Туре	Description
Boolean	A boolean expression that indicates whether the element's text shows as bold .

By default, the Bold property is True. The Bold property indicates whether the element's text shows as bold. You can use the <u>Font</u> property to change the element's font, including name, size, attributes and so on. You can use the <u>FontSize</u> property to display the element with the same font of a different size.

•		1	May	2012	2		•
	Мо	Tu	We	Th	Fr	Sa	Su
18		1	2	3	4	5	6
19	7	8	9	10	11	12	13
20	14	15	16	17	18	19	20
21	21	22	23	24	25	26	27
22	28	29	30	31			
23							

property Highlight.Font as IFontDisp

Retrieves or sets the text's font.

Туре	Description
IFontDisp	A Font object to be applied to the element.

By default, the Font property is set to nothing, which means no different font is is applied to the object. The Font property has effect only if specified a valid font. The FontSize property specifies the size of the font, with no changing of the font face.



property Highlight.FontSize as Long

Indicates the size of the font to display the text.

Туре	Description
Long	A long expression that specifies the size of the font.

By default, the FontSize property is 0, which has no effect. The Font property has effect only if specified a value different than 0. The <u>Font</u> property specifies a different font face to be applied to the element.

•		1	May	2012	2		•
	Мо	Tu	We	Th	Fr	Sa	Su
18		1	2	3	4	5	6
19	7	8	9	10	11	12	13
20	14	15	16	17	18	19	20
21	21	22	23	24	25	26	27
22	28	29	30	31			
23							

property Highlight.ForeColor as Color

Specifies the element's foreground color.

Туре	Description
Color	A Color expression that indicates the foreground color to be applied.

By default, the ForeColor property is 0. The <u>BackColor</u> property specifies the element's background color or its visual appearance using the EBN objects.

•		1	May	2012	2		•
	Мо	Tu	We	Th	Fr	Sa	Su
18		1	2	3	4	5	6
19	7	8	9	10	11	12	13
20	14	15	16	17	18	19	20
21	21	22	23	24	25	26	27
22	28	29	30	31			
23							

property Highlight.Italic as Boolean

Renders as italic text.

l

Туре	Description
Boolean	A boolean expression that indicates whether the element's text shows as <i>italic</i> .

By default, the Italic property is False. The Italic property indicates whether the element's text shows as italic. You can use the <u>Font</u> property to change the element's font, including name, size, attributes and so on. You can use the <u>FontSize</u> property to display the element with the same font of a different size.

•		1	May	2012	2		•
	Мо	Tu	We	Th	Fr	Sa	Su
18		1	2	3	4	5	6
19	7	8	9	10	11	12	13
20	14	15	16	17	18	19	20
21	21	22	23	24	25	26	27
22	28	29	30	31			
23							

property Highlight.Pattern as Pattern

Gives access to the element's Pattern object.

Туре	Description
Pattern	A Pattern object that specifies the pattern to be shown on the element.

By default, the Pattern.<u>Type</u> property is exPatternEmpty which indicates no pattern is shown.



property Highlight.StrikeOut as Boolean

Specifies that the text should appear as strikeout.

Туре	Description
Boolean	A boolean expression that indicates whether the element's text shows as strikeout .

By default, the StrikeOut property is False. The StrikeOut property indicates whether the element's text shows with a line in center. You can use the <u>Font</u> property to change the element's font, including name, size, attributes and so on. You can use the <u>FontSize</u> property to display the element with the same font of a different size.

•		1	May	2012	2		•
٦	Мо	Tu	We	Th	Fr	Sa	Su
18		1	2	3	4	5	6
19	7	8	9	10	11	12	13
20	14	45	16	47	18	19	20
21	21	22	23	24	25	26	27
22	28	29	30	31			
23							

property Highlight.Underline as Boolean

Underlines the text.

Туре	Description
Boolean	A boolean expression that indicates whether the element's text is <u>underlined</u> .

By default, the Underline property is False. The Underline property indicates whether the element's text is underlined. You can use the <u>Font</u> property to change the element's font, including name, size, attributes and so on. You can use the <u>FontSize</u> property to display the element with the same font of a different size.

•		1	May	2012	2		•
	Мо	Tu	We	Th	Fr	Sa	Su
18		1	2	3	4	5	6
19	7	8	9	10	11	12	13
20	14	15	16	17	<u>18</u>	19	20
21	21	22	23	24	<u>25</u>	<u>26</u>	27
22	28	29	30	31			
23							

MarkTime object

The MarkTime object indicates a line in the schedule view, at a specified time. The <u>Add</u> method of <u>MarkTimes</u> collection adds a new timer to the schedule view. The MarkTimes collection is accessible through the <u>MarkTimes</u> property of the control.

The MarkTime object, also called **timer**, can be used to:

- show a line of different styles on the schedule view, at specified time
- show a HTML label at specified time
- highlights the events that intersect with the timer

A time-zone (<u>MarkZone</u> object) requires the <u>Start/End</u> to define the zone, while a timer (MarkTime object) requires a <u>Time</u>, that indicates where the timer is shown.

The MarkTime object supports the following properties and methods:

Name	Description
BackColor	Specifies the background color or the visual appearance of the MarkTime object.
BodyEventBackColor	Specifies the background color or the visual appearance of the events (body) that intersect the timer.
BodyEventForeColor	Specifies the foreground color of the events (body) that intersect the timer.
BodyEventPattern	Specifies the pattern of the events (body) that intersect the timer.
<u>ForeColor</u>	Specifies the foreground color of the MarkTime object.
<u>Key</u>	Indicates the key of the marking time.
<u>Label</u>	Specifies the label to be displayed on the timer.
LabelAlign	Indicates the alignment of the timer's label.
Line	Indicates the style of the line to be shown.
LineColor	Specifies the color to show the timer's line.
Movable	Returns or sets a value that indicates whether the user can click the timer and drag it to a new position.
<u>Pattern</u>	Gives access to the element's Pattern object.
StatusEventBackColor	Specifies the background color or the visual appearance of the events (status) that intersect the timer.
<u>StatusEventForeColor</u>	Specifies the foreground color of the events (status) that intersect the timer.

<u>StatusEventPattern</u>	Specifies the pattern of the events (status) that intersect the timer.
<u>Time</u>	Specifies the date/time of the marking time.
TimeScaleBackColor	Specifies the background color or the visual appearance of the MarkTime object, in the time scale.
TimeScaleForeColor	Specifies the foreground color of the MarkTime object, in the time scale.
<u>TimeScaleLabel</u>	Specifies the label to be displayed on the timer, on the time scale part.
TimeScaleLabelAlign	Indicates the alignment of the timer's label, on the time scale part.
<u>TimeScaleLine</u>	Indicates the style of the line to be shown, on the control's time scale.
TimeScaleLineColor	Specifies the color to show the timer's line, in the time scale.
TimeScalePattern	Gives access to the element's Pattern object, being shown in the time scale.
<u>UserData</u>	Indicates any extra data associated with the MarkTime object.

property MarkTime.BackColor as Color

Specifies the background color or the visual appearance of the MarkTime object.

Туре	Description
Color	A Color expression that specifies the background color to show the timer. The last 7 bits in the high significant byte of the color to indicates the identifier of the skin being used. Use the <u>Add</u> 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.

By default, the BackColor property is 0, which means that no effect. The BackColor property has effect only, if set to a non-zero value. The BackColor property can be used to display a solid color or an EBN object on the timer's background. The BackColor property is applied to the part of the timer that shows dates section (no time scale portion of the schedule view). The Line property indicates the line to be shown. You can use the Line property on exNoLines to hide the timer's line. The <u>TimeScaleBackColor</u> property indicates the background color to be applied on the time scale portion of the control.

The following screen shot shows a timer with a different line, using the EBN object:



property MarkTime.BodyEventBackColor as Color

Specifies the background color or the visual appearance of the events (body) that intersect the timer.

Туре	Description
Color	A Color expression that specifies the background color to show the body for the events that intersect the timer. 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.

By default, the BodyEventBackColor property is 0, which means it has no effect. The BodyEventBackColor property has effect only if it is set on a non-zero value, and it changes the event's body background color (<u>BodyBackColor</u> property) for all events that intersect with the current timer. The <u>StatusEventBackColor</u> property change the event's status background color.

The following screen shot shows the events from the timer, with a different background color:



property MarkTime.BodyEventForeColor as Color

Specifies the foreground color of the events (body) that intersect the timer.

Туре	Description
Color	A Color expression that specifies the foreground color to show the body for the events that intersect the timer.

By default, the BodyEventForeColor property is 0, which means it has no effect. The BodyEventForeColor property has effect only if it is set on a non-zero value, and it changes the event's body foreground color (BodyForeColor property) for all events that intersect with the current timer. The BodyEventBackColor property changes the background color for events that intersects with the current timer. The StatusEventBackColor property change the event's status background color.

The following screen shot shows the events from the timer, with a different foreground color:



property MarkTime.BodyEventPattern as Pattern

Specifies the pattern of the events (body) that intersect the timer.

Туре	Description
Pattern	A Pattern object to specify the pattern to be displayed on the body part of the events that intersect with the current timer.

By default, the BodyEventPattern.<u>Type</u> property is exPatternEmpty which indicates that no pattern is shown. The Type property indicates the pattern to display on the status. The <u>Color</u> property indicates the color to display the pattern. The <u>FrameColor</u> property indicates the color to show the frame, if the exPatternFrame flag is included in the Type property. The <u>BodyEventBackColor</u> property change the event's body background color. The <u>StatusEventBackColor</u> property specifies the pattern of the events (status) that intersect the timer.

The following screen shot shows the events from the timer, with a different pattern, on the body part:



property MarkTime.ForeColor as Color

Specifies the foreground color of the MarkTime object.

Туре	Description
Color	A Color expression that specifies the color to show the label of the timer.

By default, the ForeColor property is 0. The ForeColor property specifies the foreground color to show the timer's Label. The Label property may include <fgcolor> HTML tag, which indicates that a portion of the label is being displayed with a different foreground color. So, if changing the timer's ForeColor is not showing any difference, please check the Label property if it contains any color attributes like <fgcolor>, <bgcolor>. The Label property specifies the label to be displayed on the timer. The LabelAlign property specifies the alignment of the label. The TimeScaleForeColor property indicates the foreground color to show the timer's TimeScaleLabel property on the time scale portion of the scheduler. You can use the <bgcolor> HTML tag to indicate a different background color for the label.

property MarkTime.Key as String

Indicates the key of the marking time.

Туре	Description
String	A String expression that specifies the key pf the timer.

The Key parameter of the <u>Add</u> method initializes the Key property of the newly created timer. The MarkTimes collection can not contain two separate MarkTime objects with the same key, so the key should be unique. If you need to change the timer's key you can remove and add a new timer with a different key. The <u>Remove</u> method removes the specified timer. The <u>Item</u> property may be used to access the timer object based on its key. The <u>MarkTimeFromPoint</u> property indicates the timer from the cursor. Calling the <u>Add</u> method with the same key, changes the <u>Time</u> property of the timer.

property MarkTime.Label as String

Specifies the label to be displayed on the timer.

Туре	Description
String	A String expression that support extended HTML format, to display a caption on the timer.

By default, the Label property is "", which indicates that no label or caption is displayed on the timer (on the dates section of the schedule view). The <u>TimeScaleLabel</u> property indicates the label to be displayed on the timer in the time scale section of the schedule view. The <u>ForeColor</u> property indicates the color to show the label, unless the <fgcolor> is not specified in the label property. The <u>LabelAlign</u> property aligns the timer's label.

Here's a few samples on how you can use the label property:

- "text", an hard coded text
- "<%hh%>:<%nn%>", hour and minute of the timer, in the 24-hours format
- "<%loc_sdate%>
<c><%hh%>:<%nn%>" displays the timer's date on the first line, while on the second line it displays the timer's time.

The property supports the following TAGs:

- <%d%> Day of the month in one or two numeric digits, as needed (1 to 31).
- <%dd%> Day of the month in two numeric digits (01 to 31).
- <%d1%> First letter of the weekday (S to S). (Use the <u>WeekDays</u> property to specify the name of the days in the week)
- <%d2%> First two letters of the weekday (Su to Sa). (Use the <u>WeekDays</u> property to specify the name of the days in the week)
- <%d3%> First three letters of the weekday (Sun to Sat). (Use the <u>WeekDays</u> property to specify the name of the days in the week)
- <%ddd%> First three letters of the weekday (Sun to Sat). (Use the <u>WeekDays</u> property to specify the name of the days in the week). You can use the <%loc_ddd%> that indicates the day of week as a three-letter abbreviation using the current user regional and language settings.
- <%loc_ddd%> Indicates the day of week as a three-letter abbreviation using the current user regional and language settings.
- <%dddd%> Full name of the weekday (Sunday to Saturday). (Use the <u>WeekDays</u> property to specify the name of the days in the week). You can use the <%loc_dddd%> that indicates day of week as its full name using the current user regional and language settings.
- <%loc_dddd%> Indicates day of week as its full name using the current user regional and language settings.

- <%i%> Displays the number instead the date. For instance, you can display numbers as 1000, 1001, 1002, 1003, instead dates. (the valid range is from -647,434 to 2,958,465)
- <%**w**%> Day of the week (1 to 7).
- <%**ww**%> Week of the year (1 to 53).
- <%**m**%> Month of the year in one or two numeric digits, as needed (1 to 12).
- <%mr%> Month of the year in Roman numerals, as needed (I to XII).
- <%mm%> Month of the year in two numeric digits (01 to 12).
- <%m1%> First letter of the month (J to D). (Use the <u>MonthNames</u> property to specify the name of the months in the year)
- <%m2%> First two letters of the month (Ja to De). (Use the <u>MonthNames</u> property to specify the name of the months in the year)
- <%m3%> First three letters of the month (Jan to Dec). (Use the <u>MonthNames</u> property to specify the name of the months in the year)
- <%mmm%> First three letters of the month (Jan to Dec). (Use the <u>MonthNames</u> property to specify the name of the months in the year). You can use the <%loc_mmm%> that indicates month as a three-letter abbreviation using the current user regional and language settings.
- <%loc_mmm%> Indicates month as a three-letter abbreviation using the current user regional and language settings.
- <%mmm%> Full name of the month (January to December). (Use the <u>MonthNames</u> property to specify the name of the months in the year). You can use the <%loc_mmm%> that indicates month as its full name using the current user regional and language settings.
- <%loc_mmmm%> Indicates month as its full name using the current user regional and language settings.
- <%q%> Date displayed as the quarter of the year (1 to 4).
- $\langle \mathbf{y} \rangle$ Number of the day of the year (1 to 366).
- <%**yy**%> Last two digits of the year (01 to 99).
- <%**yyyy**%> Full year (0100 to 9999).
- <%hy%> Date displayed as the half of the year (1 to 2).
- <%loc_gg%> Indicates period/era using the current user regional and language settings.
- <%loc_sdate%> Indicates the date in the short format using the current user regional and language settings.
- <%loc_ldate%> Indicates the date in the long format using the current user regional and language settings.
- <%loc_dsep%> Indicates the date separator using the current user regional and language settings (/).
- <%h%> Hour in one or two digits, as needed (0 to 23).
- <%**hh**%> Hour in two digits (00 to 23).
- <%**n**%> Minute in one or two digits, as needed (0 to 59).

- <%**nn**%> Minute in two digits (00 to 59).
- <%s%> Second in one or two digits, as needed (0 to 59).
- <%ss%> Second in two digits (00 to 59).
- <%AM/PM%> Twelve-hour clock with the uppercase letters "AM" or "PM", as appropriate. (Use the <u>AMPM</u> property to specify the name of the AM and PM indicators). You can use the <%loc_AM/PM%> that indicates the time marker such as AM or PM using the current user regional and language settings. You can use <%loc_A/P%> that indicates the one character time marker such as A or P using the current user regional and language settings.
- <%loc_AM/PM%> Indicates the time marker such as AM or PM using the current user regional and language settings.
- <%loc_A/P%> Indicates the one character time marker such as A or P using the current user regional and language settings.
- <%loc_time%> Indicates the time using the current user regional and language settings.
- <%loc_time24%> Indicates the time in 24 hours format without a time marker using the current user regional and language settings.
- <%loc_tsep%> indicates the time separator using the current user regional and language settings (:).

The following tags are displayed based on the user's Regional and Language Options:

- <%loc_sdate%> Indicates the date in the short format using the current user settings.
- <%loc_ldate%> Indicates the date in the long format using the current user settings.
- <%loc_ddd%> Indicates day of week as a three-letter abbreviation using the current user settings.
- <%loc_dddd%> Indicates day of week as its full name using the current user settings.
- <%loc_mmm%> Indicates month as a three-letter abbreviation using the current user settings.
- <%loc_mmmm%> Indicates month as its full name using the current user settings.
- <%loc_gg%> Indicates period/era using the current user settings.
- <%loc_dsep%> Indicates the date separator using the current user settings.
- <%loc_time%> Indicates the time using the current user settings.
- <%loc_time24%> Indicates the time in 24 hours format without a time marker using the current user settings.
- <%loc_AM/PM%> Indicates the time marker such as AM or PM using the current user settings.
- <%loc_A/P%> Indicates the one character time marker such as A or P using the current user settings.
- <%loc_tsep%> Indicates the time separator using the current user settings.
- <%loc_y%> Represents the Year only by the last digit, using current regional settings.
- <%loc_yy%> Represents the Year only by the last two digits, using current regional settings. A leading zero is added for single-digit years.
- <%loc_yyyy%> Represents the Year by a full four or five digits, depending on the calendar used. Thai Buddhist and Korean calendars have five-digit years. The "yyyy" pattern shows five digits for these two calendars, and four digits for all other supported calendars. Calendars that have single-digit or two-digit years, such as for the Japanese Emperor era, are represented differently. A single-digit year is represented with a leading zero, for example, "03". A two-digit year is represented with two digits, for example, "13". No additional leading zeros are displayed.

•

The property supports the following built-in HTML tags:

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

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

- exp, stores the plain text to be shown once the user clicks the anchor, such as "<a ;exp=show lines>"
- e64, encodes in BASE64 the HTML text to be shown once the user clicks the anchor, such as "<a ;e64=gA8ABmABnABjABvABshIAOQAEAAHAAGESikWio+ABzABohp3iELABpABu

;e64=gA8ABmABhABjABvABshIAOQAEAAHAAGESikWio+ABZABohp3iELABpABu
" that displays show lines- in gray when the user clicks the + anchor. The "gA8ABmABhABjABvABshIAOQAEAAHAAGESikWio+ABzABohp3iELABpABuABljY string encodes the "<fgcolor 808080>show lines<a>-</fgcolor>" The Decode64Text/Encode64Text methods of the eXPrint can be used to decode/encode e64 fields.

Any ex-HTML caption can be transformed to an expandable-caption, by inserting the

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

- ... displays portions of text with a different font and/or different size. For instance, the "bit" draws the bit text using the Tahoma font, on size 12 pt. If the name of the font is missing, and instead size is present, the current font is used with a different size. For instance, "bit" displays the bit text using the current font, but with a different size.
- **<fgcolor rrggbb>** ... **</fgcolor>** or **<**fgcolor=rrggbb> ... **</fgcolor>** displays text with a specified foreground color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- <bgcolor rrggbb> ... </bgcolor> or <bgcolor=rrggbb> ... </bgcolor> displays text with a specified background color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- <solidline rrggbb> ... </solidline> or <solidline=rrggbb> ... </solidline> draws a solidline 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
- number[:width] 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.
- key[:width] 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 FFFFF;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 MarkTime.LabelAlign as ContentAlignmentEnum

Indicates the alignment of the timer's label.

Туре

Description

<u>ContentAlignmentEnum</u>

A ContentAlignmentEnum expression that specifies the alignment of the timer's label.

By default, the LabelAlign property is exMiddleRight. The LabelAlign property has effect only if the <u>Label</u> property is not empty. The <u>Label</u> property indicates the caption to be displayed on the timer. The <u>TimeScaleLabelAlign</u> property aligns the timer's label to be shown in the time scale part of the schedule view.

property MarkTime.Line as LinesStyleEnum

Indicates the style of the line to be shown.

Туре	Description
<u>LinesStyleEnum</u>	A LinesStyleEnum expression that specifies the type of the line to be shown at specified time.

By default, the Line property is exSolidLine. The <u>LineColor</u> property specifies the color to display the timer's line. The <u>BackColor</u> property can be used to display a solid color or an EBN object on the timer's background. The BackColor property is applied to the part of the timer that shows dates section (no time scale portion of the schedule view). The <u>TimeScaleLine</u> property indicates the line to be shown on the time scale part of the control.

property MarkTime.LineColor as Color

Specifies the color to show the timer's line.

Туре	Description
Color	A Color expression that specifies the color to show the timer's line.

The LineColor property specifies the color to display the timer's line. The Line property indicates the style of the line to be shown by the timer. The <u>BackColor</u> property can be used to display a solid color or an EBN object on the timer's background. The BackColor property is applied to the part of the timer that shows dates section (no time scale portion of the schedule view). The <u>TimeScaleLine</u> property indicates the line to be shown on the time scale part of the control.

property MarkTime.Movable as Boolean

Returns or sets a value that indicates whether the user can click the timer and drag it to a new position.

Туре	Description
Boolean	A Boolean expression that specifies whether the user can move by dragging the timer at runtime.

By default, the Movable property is False. If the Movable property on True, the hand cursor is shown when the mouse is hovering the timer object, and the user can click and drag the timer to a new position/time. The <u>Time</u> property indicates the date/time to show the timer. The <u>AllowMoveMarkTime</u> property indicates the keys to allow user to move timers (with the Movable property on True). The <u>MarkTimeFromPoint</u> property indicates the timer from the cursor.

In conclusion, the control supports:

- fixed timers, so the user can not click and drag the timer to a new position/time, if the Movable property is False.
- movable timers, so the user can move the timer by clicking it and drag to a new position/time, if the Movable property is True.

The <u>LayoutStartChanging</u>(exScheduleMoveMarkTime) event notifies once the user is about to move a timer. The <u>MarkTimeFromPoint</u> property indicates the timer from the cursor. The <u>LayoutEndChanging</u>(exScheduleMoveMarkTime) event notifies your application once a MarkTime object is moved, at runtime.

The following snippet of code shows how to get notified once the user moves at runtime the timer. The sample holds the timer from the cursor once the <u>LayoutStartChanging(</u>exScheduleMoveMarkTime) event occurs, and when the <u>LayoutEndChanging(</u>exScheduleMoveMarkTime) event is fired, the previously saved timer, is displayed with the new time.

Dim mtChange As MarkTime

```
Private Sub Schedule1_LayoutStartChanging(ByVal Operation As
EXSCHEDULELibCtl.LayoutChangingEnum)
If (Operation = exScheduleMoveMarkTime) Then
Set mtChange = Schedule1.MarkTimeFromPoint(-1, -1)
End If
End Sub
```

Private Sub Schedule1_LayoutEndChanging(ByVal Operation As EXSCHEDULELibCtl.LayoutChangingEnum) If (Operation = exScheduleMoveMarkTime) Then If Not (mtChange Is Nothing) Then Debug.Print "Timer has been moved to " & mtChange.Time End If End If End Sub

property MarkTime.Pattern as Pattern

Gives access to the element's Pattern object.

Туре	Description
Pattern	A Pattern object to change the pattern to be displayed on the timer's background.

By default, the Pattern.<u>Type</u> property exPatternEmpty which indicates that no pattern is shown, on the timer's background. The <u>Color</u> property indicates the color to display the pattern. The <u>FrameColor</u> property indicates the color to show the frame, if the exPatternFrame flag is included in the Type property. The <u>TimeScalePattern</u> property indicates the pattern to be shown on the timer on the time scale section of schedule view.

property MarkTime.StatusEventBackColor as Color

Specifies the background color or the visual appearance of the events (status) that intersect the timer.

Color A Color expression that specifies the background color to show the status of the events that intersect the timer. 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.	Туре	Description
	Color	A Color expression that specifies the background color to show the status of the events that intersect the timer. 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.

By default, the StatusEventBackColor property is 0, which means it has no effect. The StatusEventBackColor property has effect only if it is set on a non-zero value, and it changes the event's status background color (<u>StatusColor</u> property) for all events that intersect with the current timer. The <u>StatusEventPattern</u> property specifies the pattern to be shown on the status part of the event. The <u>BodyEventBackColor</u> property change the event's body background color. The <u>ShowStatus</u> property shows or hides the status part of the event.

The following screen shot shows the events from the timer, with a different background color, for the status part:



property MarkTime.StatusEventForeColor as Color

Specifies the foreground color of the events (status) that intersect the timer.

Туре	Description
Color	A Color expression that specifies the event's foreground color.

Currently, the StatusEventForeColor property is not supported. The <u>StatusEventBackColor</u> property change the event's status background color. The <u>BodyEventBackColor</u> property change the event's body background color. The <u>ShowStatus</u> property shows or hides the status part of the event.

property MarkTime.StatusEventPattern as Pattern

Specifies the pattern of the events (status) that intersect the timer.

Туре	Description
<u>Pattern</u>	A Pattern object to specify the pattern to be displayed on the status part of the events that intersect with the current timer.

By default, the StatusEventPattern.<u>Type</u> property is exPatternEmpty which indicates that no pattern is shown. The Type property indicates the pattern to display on the status. The <u>Color</u> property indicates the color to display the pattern. The <u>FrameColor</u> property indicates the color to show the frame, if the exPatternFrame flag is included in the Type property. The <u>BodyEventBackColor</u> property change the event's body background color. The <u>ShowStatus</u> property shows or hides the status part of the event.

The following screen shot shows the events from the timer, with a different pattern, on the status part:



property MarkTime.Time as Date

Specifies the date/time of the marking time.

Туре	Description
Date	A DATE expression that specifies the date and time where the timer should be displayed.

The Time parameter of the Add method initializes the Time property. The Time property indicates the date and time to display the timer, or it indicates the position in the schedule view. You can display the timer's DATE and Time using the Label property. For instance the Label on "<%loc_sdate%> at <%hh%>:<%nn%>", displays the timer's DATE and TIME. The Movable property indicates whether the timer is moveable or fixed. The AllowMoveMarkTime property indicates the keys to allow user to move timers (with the Movable property on True). The LayoutStartChanging(exScheduleMoveMarkTime) event notifies once the user is about to move a timer. The MarkTimeFromPoint property indicates the timer from the cursor. The LayoutEndChanging(exScheduleMoveMarkTime) event notifies your application once a MarkTime object is moved, at runtime.

The following snippet of code shows how to get notified once the user moves at runtime the timer. The sample holds the timer from the cursor once the <u>LayoutStartChanging(</u> exScheduleMoveMarkTime) event occurs, and when the <u>LayoutEndChanging(</u> exScheduleMoveMarkTime) event is fired, the previously saved timer, is displayed with the new time.

Dim mtChange As MarkTime

```
Private Sub Schedule1_LayoutStartChanging(ByVal Operation As
EXSCHEDULELibCtl.LayoutChangingEnum)
If (Operation = exScheduleMoveMarkTime) Then
Set mtChange = Schedule1.MarkTimeFromPoint(-1, -1)
End If
End Sub
Private Sub Schedule1_LayoutEndChanging(ByVal Operation As
EXSCHEDULELibCtl.LayoutChangingEnum)
If (Operation = exScheduleMoveMarkTime) Then
If Not (mtChange Is Nothing) Then
Debug.Print "Timer has been moved to " & mtChange.Time
End If
End If
```

End Sub

property MarkTime.TimeScaleBackColor as Color

Specifies the background color or the visual appearance of the MarkTime object, in the time scale.

Туре	Description
Color	A Color expression that specifies the background color to show the timer, on the time scale part of the schedule view. 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.

By default, the TimeScaleBackColor property is 0, which means that no effect. The TimeScaleBackColor property has effect only, if set to a non-zero value. The TimeScaleBackColor property can be used to display a solid color or an EBN object on the timer's background. The TimeScaleBackColor property is applied on the time scale portion of the schedule view. The <u>TimeScaleLine</u> property indicates the line to be shown. You can use the TimeScaleLine property on exNoLines to hide the timer's line. The <u>BackColor</u> property indicates the background color to be applied on the dates portion of the control.

The following screen shot shows a timer with a different line, using the EBN object:



property MarkTime.TimeScaleForeColor as Color

Specifies the foreground color of the MarkTime object, in the time scale.

Туре	Description
Color	A Color expression that specifies the color to show the label of the timer in the time scale part of the schedule view.

By default, the TimeScaleForeColor property is 0. The TimeScaleForeColor property specifies the foreground color to show the timer's <u>TimeScaleLabel</u>. The TimeScaleLabel property may include <fgcolor> HTML tag, which indicates that a portion of the label is being displayed with a different foreground color. So, if changing the timer's TimeScaleForeColor is not showing any difference, please check the TimeScaleLabel property if it contains any color attributes like <fgcolor>, <bgcolor>. The TimeScaleLabel property specifies the label to be displayed on the timer, in the time scale part of the schedule view. The <u>TimeScaleLabelAlign</u> property specifies the alignment of the label. The <u>ForeColor</u> property indicates the foreground color to show the timer's <u>Label</u> property. You can use the <bgcolor> HTML tag to indicate a different background color for the label.

property MarkTime.TimeScaleLabel as String

Specifies the label to be displayed on the timer, on the time scale part.

Туре	Description
String	A String expression that support extended HTML format, to display a caption on the timer.

By default, the TimeScaleLabel property is "<fgcolor=FF0000><%hh%>:<%nn%> <%AM/PM%>", which shows the hour and the minute of the timer, using the AM/PM time indicators. The TimeScaleLabel property displays the label on the time scale portion of the schedule view. The Label property indicates the label to be displayed on the dates part of the control. The <u>TimeScaleForeColor</u> property indicates the color to show the label, unless the <fgcolor> is not specified in the label property. The <u>TimeScaleLabelAlign</u> property aligns the timer's label.

Here's a few samples on how you can use the label property:

- "text", an hard coded text
- "<%hh%>:<%nn%>", hour and minute of the timer, in the 24-hours format
- "<%loc_sdate%>
<c><%hh%>:<%nn%>" displays the timer's date on the first line, while on the second line it displays the timer's time.

The property supports the following TAGs:

- <%d%> Day of the month in one or two numeric digits, as needed (1 to 31).
- <%**dd**%> Day of the month in two numeric digits (01 to 31).
- <%d1%> First letter of the weekday (S to S). (Use the <u>WeekDays</u> property to specify the name of the days in the week)
- <%d2%> First two letters of the weekday (Su to Sa). (Use the <u>WeekDays</u> property to specify the name of the days in the week)
- <%d3%> First three letters of the weekday (Sun to Sat). (Use the <u>WeekDays</u> property to specify the name of the days in the week)
- <%ddd%> First three letters of the weekday (Sun to Sat). (Use the <u>WeekDays</u> property to specify the name of the days in the week). You can use the <%loc_ddd%> that indicates the day of week as a three-letter abbreviation using the current user regional and language settings.
- <%loc_ddd%> Indicates the day of week as a three-letter abbreviation using the current user regional and language settings.
- <%dddd%> Full name of the weekday (Sunday to Saturday). (Use the <u>WeekDays</u> property to specify the name of the days in the week). You can use the <%loc_dddd%> that indicates day of week as its full name using the current user regional and language settings.

- <%loc_dddd%> Indicates day of week as its full name using the current user regional and language settings.
- <%i%> Displays the number instead the date. For instance, you can display numbers as 1000, 1001, 1002, 1003, instead dates. (the valid range is from -647,434 to 2,958,465)
- <%**w**%> Day of the week (1 to 7).
- <%**ww**%> Week of the year (1 to 53).
- <%**m**%> Month of the year in one or two numeric digits, as needed (1 to 12).
- <%mr%> Month of the year in Roman numerals, as needed (I to XII).
- <%mm%> Month of the year in two numeric digits (01 to 12).
- <%m1%> First letter of the month (J to D). (Use the <u>MonthNames</u> property to specify the name of the months in the year)
- <%m2%> First two letters of the month (Ja to De). (Use the <u>MonthNames</u> property to specify the name of the months in the year)
- <%m3%> First three letters of the month (Jan to Dec). (Use the <u>MonthNames</u> property to specify the name of the months in the year)
- <%mmm%> First three letters of the month (Jan to Dec). (Use the <u>MonthNames</u> property to specify the name of the months in the year). You can use the <%loc_mmm%> that indicates month as a three-letter abbreviation using the current user regional and language settings.
- <%loc_mmm%> Indicates month as a three-letter abbreviation using the current user regional and language settings.
- <%mmm%> Full name of the month (January to December). (Use the <u>MonthNames</u> property to specify the name of the months in the year). You can use the <%loc_mmm%> that indicates month as its full name using the current user regional and language settings.
- <%loc_mmmm%> Indicates month as its full name using the current user regional and language settings.
- <%q%> Date displayed as the quarter of the year (1 to 4).
- $\langle \mathbf{y} \mathbf{y} \rangle$ Number of the day of the year (1 to 366).
- <%yy%> Last two digits of the year (01 to 99).
- <%**yyyy**%> Full year (0100 to 9999).
- <%hy%> Date displayed as the half of the year (1 to 2).
- <%loc_gg%> Indicates period/era using the current user regional and language settings.
- <%loc_sdate%> Indicates the date in the short format using the current user regional and language settings.
- <%loc_ldate%> Indicates the date in the long format using the current user regional and language settings.
- <%loc_dsep%> Indicates the date separator using the current user regional and language settings (/).
- <%h%> Hour in one or two digits, as needed (0 to 23).

- <%**hh**%> Hour in two digits (00 to 23).
- <%n%> Minute in one or two digits, as needed (0 to 59).
- <%**nn**%> Minute in two digits (00 to 59).
- <%**s**%> Second in one or two digits, as needed (0 to 59).
- <%ss%> Second in two digits (00 to 59).
- <%AM/PM%> Twelve-hour clock with the uppercase letters "AM" or "PM", as appropriate. (Use the <u>AMPM</u> property to specify the name of the AM and PM indicators). You can use the <%loc_AM/PM%> that indicates the time marker such as AM or PM using the current user regional and language settings. You can use <%loc_A/P%> that indicates the one character time marker such as A or P using the current user regional and language settings.
- <%loc_AM/PM%> Indicates the time marker such as AM or PM using the current user regional and language settings.
- <%loc_A/P%> Indicates the one character time marker such as A or P using the current user regional and language settings.
- <%loc_time%> Indicates the time using the current user regional and language settings.
- <%loc_time24%> Indicates the time in 24 hours format without a time marker using the current user regional and language settings.
- <%loc_tsep%> indicates the time separator using the current user regional and language settings (:).

The following tags are displayed based on the user's Regional and Language Options:

- <%loc_sdate%> Indicates the date in the short format using the current user settings.
- <%loc_ldate%> Indicates the date in the long format using the current user settings.
- <%loc_ddd%> Indicates day of week as a three-letter abbreviation using the current user settings.
- <%loc_dddd%> Indicates day of week as its full name using the current user settings.
- <%loc_mmm%> Indicates month as a three-letter abbreviation using the current user settings.
- <%loc_mmmm%> Indicates month as its full name using the current user settings.
- <%loc_gg%> Indicates period/era using the current user settings.
- <%loc_dsep%> Indicates the date separator using the current user settings.
- <%loc_time%> Indicates the time using the current user settings.
- <%loc_time24%> Indicates the time in 24 hours format without a time marker using the current user settings.
- <%loc_AM/PM%> Indicates the time marker such as AM or PM using the current user settings.
- <%loc_A/P%> Indicates the one character time marker such as A or P using the

current user settings.

- <%loc_tsep%> Indicates the time separator using the current user settings.
- <%loc_y%> Represents the Year only by the last digit, using current regional settings.
- <%loc_yy%> Represents the Year only by the last two digits, using current regional settings. A leading zero is added for single-digit years.
- <%loc_yyyy%> Represents the Year by a full four or five digits, depending on the calendar used. Thai Buddhist and Korean calendars have five-digit years. The "yyyy" pattern shows five digits for these two calendars, and four digits for all other supported calendars. Calendars that have single-digit or two-digit years, such as for the Japanese Emperor era, are represented differently. A single-digit year is represented with a leading zero, for example, "03". A two-digit year is represented with two digits, for example, "13". No additional leading zeros are displayed.

The property supports the following built-in HTML tags:

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

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

- exp, stores the plain text to be shown once the user clicks the anchor, such as "<a ;exp=show lines>"

;e64=gA8ABmABnABjABvABshIAOQAEAAHAAGESikWio+ABzABohp3iELABpABu " that displays show lines- in gray when the user clicks the + anchor. The "gA8ABmABnABjABvABshIAOQAEAAHAAGESikWio+ABzABohp3iELABpABuABljY string encodes the "<fgcolor 808080>show lines<a>-</fgcolor>" The Decode64Text/Encode64Text methods of the eXPrint can be used to decode/encode e64 fields. Any ex-HTML caption can be transformed to an expandable-caption, by inserting the anchor ex-HTML tag. For instance, "<solidline>Header</solidline>
Line1<r><a ;exp=show lines>+
Line2
Line3" shows the Header in underlined and bold on the first line and Line1, Line2, Line3 on the rest. The "show lines" is shown instead of Line1, Line2, Line3 once the user clicks the + sign.

- ... displays portions of text with a different font and/or different size. For instance, the "bit" draws the bit text using the Tahoma font, on size 12 pt. If the name of the font is missing, and instead size is present, the current font is used with a different size. For instance, "bit" displays the bit text using the current font, but with a different size.
- **<fgcolor rrggbb>** ... **</fgcolor>** or **<**fgcolor=rrggbb> ... **</**fgcolor> displays text with a specified foreground color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- <bgcolor rrggbb> ... </bgcolor> or <bgcolor=rrggbb> ... </bgcolor> displays text with a specified background color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- <solidline rrggbb> ... </solidline> or <solidline=rrggbb> ... </solidline> draws a solidline 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
- number[:width] 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.
- key[:width] 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 FFFFF;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=FFFFF>outline anti-aliasing</fgcolor> </**sha**>" gets:

outline anti-aliasing

property MarkTime.TimeScaleLabelAlign as ContentAlignmentEnum

Indicates the alignment of the timer's label, on the time scale part.

TypeDescriptionContentAlignmentEnumA ContentAlignmentEnum expression that specifies the
alignment of the timer's label.

By default, the TimeScaleLabelAlign property is exMiddleCenter. The TimeScaleLabelAlign property has effect only if the <u>TimeScaleLabel</u> property is not empty. The <u>TimeScaleLabel</u> property indicates the caption to be displayed on the timer, in the time scale part of the schedule view. The <u>LabelAlign</u> property aligns the timer's label to be shown in the dates part of the schedule view.

property MarkTime.TimeScaleLine as LinesStyleEnum

Indicates the style of the line to be shown, on the control's time scale.

Туре	Description
<u>LinesStyleEnum</u>	A LinesStyleEnum expression that specifies the type of the line to be shown at specified time.

By default, the TimeScaleLine property is exSolidLine. The <u>TimeScaleLineColor</u> property specifies the color to display the timer's line. The <u>TimeScaleBackColor</u> property can be used to display a solid color or an EBN object on the timer's background, on the time scale portion of the schedule view. The <u>Line</u> property indicates the line to be shown on the dates part of the control.

property MarkTime.TimeScaleLineColor as Color

Specifies the color to show the timer's line, in the time scale.

Туре	Description
Color	A Color expression that specifies the color to show the timer's line.

The TimeScaleLineColor property specifies the color to display the timer's line. The <u>TimeScaleLine</u> property indicates the style of the line to be shown by the timer. The <u>TimeScaleBackColor</u> property can be used to display a solid color or an EBN object on the timer's background. The TimeScaleBackColor property is applied to the part of the timer that shows in time scale of the schedule section. The <u>Line</u> property indicates the line to be shown on the time scale part of the control.

property MarkTime.TimeScalePattern as Pattern

Gives access to the element's Pattern object, being shown in the time scale.

Туре	Description
Pattern	A Pattern object to change the pattern to be displayed on the timer's background.

By default, the TimeScalePattern.Type property exPatternEmpty which indicates that no pattern is shown, on the timer's background. The <u>Color</u> property indicates the color to display the pattern. The <u>FrameColor</u> property indicates the color to show the frame, if the exPatternFrame flag is included in the Type property. The <u>Pattern</u> property indicates the pattern to be shown on the timer on dates section of schedule view.

property MarkTime.UserData as Variant

Indicates any extra data associated with the MarkTime object.

Туре	Description
Variant	A VARIANT expression that specifies any extra data associated with the timer object.

By default, the UserData property is empty or nothing. You can use the UserData property to associate or assign any extra data to the timer. The <u>Label</u> property specifies the label to be shown on the timer.

MarkTimes object

The MarkTimes collection holds <u>MarkTime</u> objects, also called timers. The MarkTimes collection is accessible through the <u>MarkTimes</u> property of the control.

The MarkTime object, also called timer, can be used to:

- show a line of different styles on the schedule view, at specified time
- show a HTML label at specified time
- highlights the events that intersect with the timer

The following sample shows a timer on the control:

	Thu	rsday, May 24	4, 2012
08:00 AM :30			
09:00 AM :30	9:00 AM -		
10:00 AM :30			
11:00 AM :30		10:45 AM - 12:45 PM	
12:00 PM	12:38 PM		2:30 PM
01:00 PM :30	12:45 PM - 3	:45 PM	
02:00 PM :30			
03:00 PM :30			

The MarkTimes collection supports the following properties and methods:

Name	Description
Add	Adds a MarkTime object to the collection and returns a reference to the newly created object.
<u>Clear</u>	Removes all objects in a collection.
<u>Count</u>	Returns the number of objects in a collection.
<u>ltem</u>	Returns a specific MarkTime of the MarkTimes collection, giving its key.
Remove	Removes a specific member from the MarkTimes collection.

method MarkTimes.Add (Key as String, Time as Date)

Adds a MarkTime object to the collection and returns a reference to the newly created object.

Туре	Description
Key as String	A String expression that specifies the key of the timer to be added. Once the newly timer is created, the <u>Key</u> property indicates the key of the timer.
Time as Date	A DATE expression that specifies the date/time where the timers should be displayed. The <u>Time</u> property indicates the date/time where the timer is shown.
Return	Description
<u>MarkTime</u>	A MarkTime object being created.

The Add method of <u>MarkTimes</u> collection adds a new timer to the schedule view. The <u>Time</u> property indicates the date/time where the timer is shown. The MarkTimes collection is accessible through the <u>MarkTimes</u> property of the control. The <u>ShowMarkTime</u> property indicates whether the schedule view displays timers. The <u>AllowMoveMarkTime</u> property indicates the keys to allow user to move timers (with the <u>Movable</u> property on True). The <u>MarkTimeFromPoint</u> property indicates the timer from the cursor.

The MarkTime object, also called **timer**, can be used to:

- show a line of different styles on the schedule view, at specified time
- show a HTML label at specified time
- highlights the events that intersect with the timer

The following screen shot shows the control's timers (red and black/blue arrow):

Tuesday, May 15, 2012	Wednesday, May 16, 2012		-
Tuesday, May 15, 2012	Wednesday, May 16, 2012 8:40 AM - 11:10 AM 11:10 AM - 3:20 PM 11:20 AM - 12:10 PM 12:10 PM - 3:00 PM 5/16/20	08:00 AM 20 40 09:00 AM 20 40 10:00 AM 20 10:00 AM 20 11:00 AM 20 40 11:00 AM 20 40 11:00 PM 20 40 12:00 PM 20 40 12:00 PM 20 40 12:00 PM 20 40 12:00 AM 20 40 10:00 AM 20 10:00 AM 10:00 AM 20 10:00 AM 20 10:00 AM 20 10:00 AM 10:00 AM 20 10:00 AM 10:00	
		20 :40 03:00 PM :20 :40	

method MarkTimes.Clear ()

Removes all objects in a collection.

Туре

Description

The Clear method release all added timers. The Clear method remove all timers objects. You can use <u>ShowMarkTime</u> property to show or hide the schedule's timers. The <u>Remove</u> method removes a specific timer. The <u>AllowMoveMarkTime</u> property indicates the keys to allow user to move timers (with the <u>Movable</u> property on True).

property MarkTimes.Count as Long

Returns the number of objects in a collection.

l

Туре	Description
Long	A Long expression that specifies the number of MarkTime objects in the control.

The Count property gets the number of the MarkTime objects in the MarkTimes collection. The <u>Item</u> property accesses a MarkTime object based on its index. The <u>AllowMoveMarkTime</u> property indicates the keys to allow user to move timers (with the <u>Movable</u> property on True). The <u>MarkTimeFromPoint</u> property indicates the timer from the cursor. The **for each statement** is supported by the MarkTimes collection, so the MarkTimes collection can be enumerated using a sample like *for each mt in MarkTimes*

property MarkTimes.Item (Key as Variant) as MarkTime

Returns a specific MarkTime of the MarkTimes collection, giving its key.

Туре	Description
Key as Variant	A Key expression that indicates the key of the timer object being requested.
<u>MarkTime</u>	A MarkTime object, called timer, to be requested, or empty/nothing/NULL, if no timer with specified key has been found.

The Item property gives the timer object based on its key. The <u>Count</u> property gets the number of the MarkTime objects in the MarkTimes collection. The <u>Add</u> method of <u>MarkTimes</u> collection adds a new timer to the schedule view, with a specified key. The <u>ShowMarkTime</u> property indicates whether the schedule view displays timers. The <u>AllowMoveMarkTime</u> property indicates the keys to allow user to move timers (with the <u>Movable</u> property on True). The <u>MarkTimeFromPoint</u> property indicates the timer from the cursor. The **for each statement** is supported by the MarkTimes collection, so the MarkTimes collection can be enumerated using a sample like *for each mt in MarkTimes*

method MarkTimes.Remove (Key as Variant)

Removes a specific member from the MarkTimes collection.

Туре	Description
Key as Variant	A String expression that specifies the key of the timer to be removed.

The Remove method removes a specific timer. The <u>Clear</u> method remove all timers objects. You can use <u>ShowMarkTime</u> property to show or hide the schedule's timers. The <u>AllowMoveMarkTime</u> property indicates the keys to allow user to move timers (with the <u>Movable</u> property on True). The <u>MarkTimeFromPoint</u> property indicates the timer from the cursor.
MarkZone object

A MarkZone object holds information about a time-zone. A time-zone is identified by a <u>Start/End</u> date time, what can be highlighted in the schedule view. The <u>MarkZones</u> collection contains a collection of the MarkZone objects. The MarkZones collection is accessible through the <u>MarkZones</u> property of the control.

The MarkZone object can:

- show a HTML caption on a specified time-zone
- highlight a time-zone with a different background, pattern and so on, to indicate a restricted zone for instance.

A time-zone (MarkZone object) requires the <u>Start/End</u> to define the zone, while a timer (<u>MarkTime</u> object) requires a <u>Time</u>, that indicates where the timer is shown.

The MarkZone object supports the following properties and methods:

Name	Description
BackColor	Specifies the background color or the visual appearance of the MarkZone object.
End	Specifies the ending date/time of the marking zone.
<u>ForeColor</u>	Specifies the foreground color of the MarkZone object.
<u>GroupID</u>	Specifies the identifier of the group where the MarkZone object should be shown.
<u>Key</u>	Indicates the key of the marking zone.
<u>LongLabel</u>	Specifies the long label to be displayed on the marking zone.
Pattern	Specifies the pattern to show the MarkZone object
<u>ShortLabel</u>	Specifies the short label to be displayed on the marking zone.
<u>Start</u>	Specifies the starting date/time of the marking zone.

property MarkZone.BackColor as Color

Specifies the background color or the visual appearance of the MarkZone object.

Туре	Description
Color	A Color expression that specifies the background color to show the time-zone. The last 7 bits in the high significant byte of the color to indicates the identifier of the skin being used. Use the <u>Add</u> 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.

By default, the BackColor property is 0, which means that no effect. The BackColor property has effect only, if set to a non-zero value. The BackColor property can be used to display a solid color or an EBN object on the time-zone's background. The <u>Color</u> property indicates the color to display the pattern, while the <u>Pattern. Type</u> property is not exPatternEmpty. The <u>Pattern.FrameColor</u> property indicates the color to show the frame, if the exPatternFrame flag is included in the <u>Pattern.Type</u> property.

The following screen shot shows the time-zone on all groups:



property MarkZone.End as Date

Specifies the ending date/time of the marking zone.

Туре	Description
Date	A DATE expression that specifies the ending point of the time-zone.

The time-zone is identified on the schedule view by the <u>Start</u> (starting point) and End (ending point). The End parameter of the <u>Add</u> method initializes the End property of the time-zone. You can use the End property to programmatically extend the time-zone. The Start should be less than End property, else the time-zone is not visible.

property MarkZone.ForeColor as Color

Specifies the foreground color of the MarkZone object.

Туре	Description
Color	A Color expression that specifies the color to show the label of the time-zone.

By default, the ForeColor property is 0. The ForeColor property specifies the foreground color to show the time-zone's <u>ShortLabel/LongLabel</u>. The label property may include <fgcolor> HTML tag, which indicates that a portion of the label is being displayed with a different foreground color. So, if changing the time-zone's ForeColor is not showing any difference, please check the Label property if it contains any color attributes like <fgcolor>, <bgcolor>. The Label property specifies the label to be displayed on the time-zone.

property MarkZone.GroupID as Long

Specifies the identifier of the group where the MarkZone object should be shown.

Туре	Description
Long	A long expression that specifies the identifier of the group where the time-zone should be shown.

By default, the GroupID property is -1, which means that the time-zone is not assigned to any group. Use a positive value to assign the time-zone to a specific group. Use the Add method of the Groups collection to add new groups to the control. The <u>DisplayGroupingButton</u> property indicates whether the header of the date displays the grouping button. The list of available groups is displayed on a drop down panel, once the user clicks the grouping/filtering button. The <u>GroupID</u> property specifies the identifier of the group where the event belongs. If the control displays groups the GroupID property of the newly created event is automatically updated with the group where the event has been created. The <u>AllowMoveEventToOtherGroup</u> property specifies whether the user can move an event from a group to another at runtime.

The control displays groups if:

- <u>ShowGroupingEvents</u> property is True
- The <u>Groups</u> collection has elements. By default, the Groups collection contains no Group objects.
- The Groups collection contains visible groups (<u>Visible</u> property on True)

The following screen shot shows time-zones on different groups:



property MarkZone.Key as String

Indicates the key of the marking zone.

Туре	Description
String	A String expression that specifies the key of the time- zone.

The Key parameter of the <u>Add</u> method initializes the Key property of the newly created time-zone. The MarkZones collection can not contain two separate MarkZone objects with the same key, so the key should be unique. If you need to change the time-zone's key you can remove and add a new time-zone with a different key. The <u>Remove</u> method removes the specified time-zone. The <u>Item</u> property may be used to access the time-zone object based on its key. The <u>MarkZoneFromPoint</u> property indicates the time-zone from the cursor.

property MarkZone.LongLabel as String

Specifies the long label to be displayed on the marking zone.

Туре	Description
String	A String expression that specifies the HTML caption to be displayed on the time-zone.

By default, the LongLabel property is "", which indicates no caption is being displayed on the time-zone. The time-zone may display the <u>ShortLabel</u> or LongLabel. The LongLabel of the time-zone is shown if its width fits the time-zone's width, else the ShortLabel property is displayed. The LongLabel property supports HTML format. You can use the
 HTML tag to print the caption on multiple lines. You can use the HTML tag to display icons or pictures on the time-zone. The <u>ForeColor</u> property indicates the foreground color to show the time-zone's label, unless the <fgcolor> HTML tag is not used in the label property.

Here's a few samples on how you can use the label property:

- "text", an hard coded text
- "pic1
<c>text", displays the a picture on the first line, and text on the second line.

The property supports the following built-in HTML tags:

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

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

- exp, stores the plain text to be shown once the user clicks the anchor, such as "<a ;exp=show lines>"
- e64, encodes in BASE64 the HTML text to be shown once the user clicks the anchor, such as "<a/li>

;e64=gA8ABmABnABjABvABshIAOQAEAAHAAGESikWio+ABzABohp3iELABpABu " that displays show lines- in gray when the user clicks the + anchor. The "gA8ABmABnABjABvABshIAOQAEAAHAAGESikWio+ABzABohp3iELABpABuABljY string encodes the "<fgcolor 808080>show lines<a>-</fgcolor>" The Decode64Text/Encode64Text methods of the eXPrint can be used to decode/encode e64 fields.

Any ex-HTML caption can be transformed to an expandable-caption, by inserting the anchor ex-HTML tag. For instance, "<solidline>Header</solidline>

Line1<r><a ;exp=show lines>+
Line2
Line3" shows the Header in underlined and bold on the first line and Line1, Line2, Line3 on the rest. The "show lines" is shown instead of Line1, Line2, Line3 once the user clicks the + sign.

- ... displays portions of text with a different font and/or different size. For instance, the "bit" draws the bit text using the Tahoma font, on size 12 pt. If the name of the font is missing, and instead size is present, the current font is used with a different size. For instance, "bit" displays the bit text using the current font, but with a different size.
- **<fgcolor rrggbb> ... </fgcolor>** or <fgcolor=rrggbb> ... </fgcolor> displays text with a specified foreground color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- <bgcolor rrggbb> ... </bgcolor> or <bgcolor=rrggbb> ... </bgcolor> displays text with a specified background color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- <solidline rrggbb> ... </solidline> or <solidline=rrggbb> ... </solidline> draws a solidline 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
- number[:width] 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 <u>Add</u> 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.

- key[:width] 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 **<**font face;size> to define a smaller or a larger font to be displayed. For instance: "Text with **<**font ;7>**<off** 6>subscript" displays the text

such as: Text with _{subscript} The "Text with <**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 FFFFF;1;1>gradient-center</gra>" generates the following picture:

gradient-center

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

outlined

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

the font. For instance the "<**sha**>shadow</**sha**>" generates the following picture:

shadow

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



property MarkZone.Pattern as Pattern

Specifies the pattern to show the MarkZone object

Туре	Description
Pattern	A Pattern object to change the pattern to be displayed on the time-zone's background.

By default, the Pattern.<u>Type</u> property exPatternHorizontal. Use the Pattern.<u>Type</u> property to change the pattern to be displayed on the time-zone's background. The <u>Color</u> property indicates the color to display the pattern. The <u>FrameColor</u> property indicates the color to show the frame, if the exPatternFrame flag is included in the Type property.

property MarkZone.ShortLabel as String

Specifies the short label to be displayed on the marking zone.

Туре	Description
String	A String expression that specifies the caption to be displayed on the time-zone.

By default, the ShortLabel property is "", which indicates no caption is being displayed on the time-zone. The time-zone may display the ShortLabel or LongLabel. The LongLabel of the time-zone is shown if its width fits the time-zone's width, else the ShortLabel property is displayed. The ShortLabel property displays no HTML format. The ForeColor property indicates the foreground color to show the time-zone's label, unless the <fgcolor> HTML tag is not used in the label property.

property MarkZone.Start as Date

Specifies the starting date/time of the marking zone.

Туре	Description
Date	A DATE expression that specifies the starting point of the time-zone.

The time-zone is identified on the schedule view by the Start (starting point) and <u>End</u> (ending point). The Start parameter of the <u>Add</u> method initializes the Start property of the time-zone. You can use the Start property to programmatically extend the time-zone. The Start should be less than End property, else the time-zone is not visible.

MarkZones object

The MarkZones collection holds a collection of MarkZone objects, also called time-zone.

The MarkZone object can:

- show a HTML/Image caption on a specified time-zone
- highlight a time-zone with a different background, pattern and so on, to indicate a restricted zone for instance.

A time-zone (MarkZone object) requires the <u>Start/End</u> to define the zone, while a timer (<u>MarkTime</u> object) requires a <u>Time</u>, that indicates where the timer is shown.

The following screen shot shows a time-zone on the control:



The MarkZones collection supports the following properties and methods:

Name	Description
Add	Adds a MarkZone object to the collection and returns a reference to the newly created object.
<u>Clear</u>	Removes all objects in a collection.
<u>Count</u>	Returns the number of objects in a collection.
<u>ltem</u>	Returns a specific MarkZone of the MarkZones collection, giving its key.
	Removes a specific member from the MarkZones

method MarkZones.Add (Key as String, Start as Variant, End as Variant)

Adds a MarkZone object to the collection and returns a reference to the newly created object.

Туре	Description
Key as String	A String expression that indicates the key of the time-zone to be added. The Key property of the MarkZone object indicates the key of the newly added time-zone.
Start as Variant	A DATE expression that specifies the starting point of the time-zone. The <u>Start</u> property indicates the starting point of the time-zone.
End as Variant	A DATE expression that specifies the ending point of the time-zone. The <u>End</u> property indicates the ending point of the time-zone.
Return	Description
MarkZone	A MarkZone object being added.

Use the Add method of the MarkZones collection to add a new time-zone to the control. The <u>Start</u> property indicates the starting date/time of the time-zone. The <u>End</u> property indicates the starting date/time of the time-zone. The <u>MarkZones</u> property gets the MarkZones collection. The MarkZones collection holds a set of <u>MarkZone</u> objects (also called time-zone). A MarkZone object holds information about a time-zone. A time-zone is identified by a <u>Start/End</u> date time, what can be highlighted in the schedule view. The <u>MarkZoneFromPoint</u> property indicates the time-zone from the cursor. The Start should be less than End property, else the time-zone is not visible.

The MarkZone object can:

- show a HTML/Image caption on a specified time-zone
- highlight a time-zone with a different background, pattern and so on, to indicate a restricted zone for instance.

A time-zone (<u>MarkZone</u> object) requires the <u>Start/End</u> to define the zone, while a timer (<u>MarkTime</u> object) requires a <u>Time</u>, that indicates where the timer is shown.

method MarkZones.Clear ()

Removes all objects in a collection.

Туре

Description

The Clear method removes all elements in the MarkZones collection, or clears all timezones. The <u>ShowMarkZone</u> property shows or hides the added time-zones. The <u>Remove</u> method removes a specific time-zone. The <u>MarkZoneFromPoint</u> property indicates the timezone from the cursor. Use the <u>Add</u> method of the MarkZones collection to add a new timezone to the control.

property MarkZones.Count as Long

Returns the number of objects in a collection.

Туре	Description
Long	A Long expression that specifies the number of time- zones.

The Count property indicates the number of MarkZone objects in the MarkZones collection. The <u>Item</u> property gets a time-zone based on its key. The <u>Start</u> property indicates the starting date/time of the time-zone. The <u>End</u> property indicates the starting date/time of the time-zone. The **for each statement** is supported by the MarkZones collection, so the MarkZones collection can be enumerated using a sample like *for each mz in MarkZones*

property MarkZones.Item (Key as Variant) as MarkZone

Returns a specific MarkZone of the MarkZones collection, giving its key.

Туре	Description
Key as Variant	A String expression that specifies the key of the time-zone to be accessed.
<u>MarkZone</u>	A MarkZone object associated with the specified key, or empty/nothing/NULL if no time-zone associated to the giving key.

The Item property gets a time-zone based on its key. The <u>Count</u> property indicates the number of MarkZone objects in the MarkZones collection. The <u>Start</u> property indicates the starting date/time of the time-zone. The <u>End</u> property indicates the starting date/time of the time-zone. The <u>End</u> property indicates the starting date/time of the time-zone. The <u>End</u> property indicates the starting date/time of the MarkZones collection, so the MarkZones collection can be enumerated using a sample like *for each mz in MarkZones*

method MarkZones.Remove (Key as Variant)

Removes a specific member from the MarkZones collection.

Туре	Description
Key as Variant	A String expression that specifies the key of the time-zone to be removed.

The Remove method removes a specific time-zone. The <u>MarkZoneFromPoint</u> property indicates the time-zone from the cursor. The <u>ShowMarkZone</u> property shows or hides the added time-zones. The <u>Clear</u> method removes all elements in the MarkZones collection, or clears all time-zones. Use the <u>Add</u> method of the MarkZones collection to add a new time-zone to the control.

NonworkingPattern object

The NonworkingPattern object holds the colors/pattern to display a non-working timeinterval. The <u>NonworkingTime</u> object defines the non-working time-interval. Each <u>NonworkingTime</u> object can associate a NonworkingPattern object to define the colors/pattern to display the non-working time-interval. The <u>ID</u> property of NonworkingPattern defines the identifier of the pattern/color. The <u>IDNonworkingPattern</u> property of the <u>NonworkingTime</u> object indicates the identifier of the NonworkingPattern object to be displayed on the non-working interval.

The NonworkingPattern object supports the following properties and methods:

Name	Description
BackColor	Specifies the non-working's background color.
BackgroundExt	Indicates additional colors, text, images that can be displayed on the pattern's background using the EBN string format.
ID	Gets identifier of the current NonworkingPattern object.
Pattern	Specifies the pattern to show the non-working part

property NonworkingPattern.BackColor as Color

Specifies the non-working's background color.

Туре	Description
Color	A Color expression that specifies the background color to show the non-working time-interval. 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.

By default, the BackColor property is 0, which means that no effect. The BackColor property has effect only, if set to a non-zero value. The BackColor property can be used to display a solid color or an EBN object on the non-working time-interval's background. The <u>Pattern.Color</u> property indicates the color to display the pattern, while the <u>Pattern.Type</u> property is not exPatternEmpty. The <u>Pattern.FrameColor</u> property indicates the color to show the frame, if the exPatternFrame flag is included in the <u>Pattern.Type</u> property. The <u>BackgroundExt</u> property specifies additional colors, text, images that can be displayed on the pattern's background using the EBN string format.

property NonworkingPattern.BackgroundExt as String

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

Type Des	cription
A St	tring expression ("EBN String Format") that defines
the	layout of the UI to be applied on the object's
bac	kground. The <u>syntax</u> of EBN String Format in BNF
nota	ation is shown bellow. You can use the EBN's Builder
of e	XButton/COM control to define visually the EBN
Strin	ing 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 non-working time-interval's background. The <u>BackColor</u> property can be used to display a solid color or an EBN object on the non-working time-interval's background.

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=`flag`,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 <u>BDiagonal</u> pattern with a border arround on the lower-half part of the object's background.



 "root[text=`caption 2`,align=0x22](client[text=`caption 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 <u>eXButton</u> 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 BodyBackgroundExt property.

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

```
<EBN> ::= <elements> | <root> "(" [<elements>] ")"
<elements> ::= <element> [ "," <elements> ]
<root> ::= "root" [ <attributes> ] | [ <attributes> ]
<element> ::= <anchor> [ <attributes> ] [ "(" [<elements>] ")" ]
<anchor> ::= "none" | "left" | "right" | "client" | "top" | "bottom"
```

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

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

property NonworkingPattern.ID as Long

Gets identifier of the current NonworkingPattern object.

Туре	Description
Long	A long expression that defines the identifier of the colors/pattern to be used by a non-working time-interval.

The ID property is a a read-only property. The ID property is initialized by the ID parameter of the <u>Add</u> method. The <u>IDNonworkingPattern</u> property of the <u>NonworkingTime</u> object indicates the identifier of the NonworkingPattern object to be displayed on the non-working interval. In order to change the ID property, you can remove the object and add a new NonworkingPattern object using the Add method with a different identifier. The <u>Remove</u> method removes a NonworkingPattern object based on identifier.

property NonworkingPattern.Pattern as Pattern

Specifies the pattern to show the non-working part

l

Туре	Description
Pattern	A Pattern object that specifies the pattern to be displayed on a non-working time-interval.

The the <u>Pattern. Type</u> property is initialized with the value of Type property of the <u>Add</u> method. The <u>BackColor</u> property can be used to display a solid color or an EBN object on the non-working time-interval's background. The <u>Pattern.Color</u> property indicates the color to display the pattern, while the <u>Pattern. Type</u> property is not exPatternEmpty. The <u>Pattern.FrameColor</u> property indicates the color to show the frame, if the exPatternFrame flag is included in the <u>Pattern.Type</u> property. The <u>BackgroundExt</u> property specifies additional colors, text, images that can be displayed on the pattern's background using the EBN string format.

NonworkingPatterns object

The NonworkingPatterns collection holds a list of <u>NonworkingPattern</u> objects. The NonworkingPatterns collection defines the patterns/colors to be used by non-working time intervals. In other words, each NonworkingTime object can associate a <u>NonworkingPattern</u> object that specifies the colors and the pattern to show the non-working zone. The <u>ShowNonworkingTime</u> property shows or hides the defined non-working intervals. The <u>NonworkingTimeFromPoint</u> property gets the non-working object from the cursor. The NonworkingPatterns collection is accessible through the <u>NonworkingPatterns</u> property of the control.

The <u>ID</u> property defines the identifier of the pattern/color. The <u>IDNonworkingPattern</u> property indicates the identifier of the NonworkingPattern object to be displayed on the non-working interval.

The <u>NonworkingDays</u> property of the calendar defines the days to be non-working in the calendar. The <u>NonworkingDaysPattern</u> and the <u>NonworkingDaysColor</u> which defines the pattern and the color to show the non-working days. The <u>FirstWeekDay</u> property indicates the first day of the week. The <u>Add</u> method of the NonworkingTimes objects adds a new non-working time interval.

The NonworkingPatterns collection supports the following properties and methods:

Name	Description
Add	Adds a NonworkingPattern object to the collection and returns a reference to the newly created object.
<u>Clear</u>	Removes all objects in a collection.
<u>Count</u>	Returns the number of objects in a collection.
<u>ltem</u>	Returns a specific NonworkingPattern of the NonworkingPatterns collection, giving its identifier.
<u>Remove</u>	Removes a specific member from the NonworkingPatterns collection.

method NonworkingPatterns.Add (ID as Long, Type as PatternEnum)

Adds a NonworkingPattern object to the collection and returns a reference to the newly created object.

Туре	Description
ID as Long	A Long expression that defines the identifier of the colors/pattern to be used to show a non-working time-interval.
Type as <u>PatternEnum</u>	A PatternEnum expression specifies the type of the pattern to be assigned to non-working time-interval.
Return	Description
NonworkingPattern	A NonworkingPattern object being created.

The Add method adds a color/pattern to be displayed on a non-working time-interval. The ID property of the NonworkingTime object defines the identifier of the pattern/color. The IDNonworkingPattern property indicates the identifier of the NonworkingPattern object to be displayed on the non-working interval. Each NonworkingTime object can associate a NonworkingPattern object that specifies the colors and the pattern to show the non-working intervals. The ShowNonworkingTime property shows or hides the defined non-working intervals. The NonworkingTimeFromPoint property gets the non-working object from the cursor. The NonworkingPatterns collection is accessible through the NonworkingPatterns property of the control. The Add method of the NonworkingTimes objects adds a new non-working time interval.

The NonworkingTime's advantages are:

- highlight the interval of time as non-working with a different patterns, colors.
- any/all day can display different intervals of time as non-working
- you can specify the non-working interval using an expression, that defines the days where the non-working interval is shown.

method NonworkingPatterns.Clear ()

Removes all objects in a collection.

Туре

Description

The Clear method removes all defined <u>NonworkingPattern</u> objects. The <u>Remove</u> method removes a NonworkingPattern object based on its identifier. The <u>Count</u> property counts the number of NonworkingPattern objects in the NonworkingPatterns collection. The <u>Item</u> property accesses a NonworkingPattern object based on its identifier. The <u>ID</u> property indicates the identifier of the NonworkingPattern object. The <u>IDNonworkingPattern</u> property of the <u>NonworkingTime</u> object indicates the identifier of the NonworkingPattern object to be displayed on the non-working interval.

property NonworkingPatterns.Count as Long

Returns the number of objects in a collection.

Туре	Description
Long	A long expression that defines the number of the NonworkingPattern objects in the NonworkingPatterns collection.

The Count property counts the number of NonworkingPattern objects in the NonworkingPatterns collection. The Item property accesses a NonworkingPattern object based on its identifier. The ID property indicates the identifier of the NonworkingPattern object. The IDNonworkingPattern property of the NonworkingTime object indicates the identifier of the NonworkingPattern object to be displayed on the non-working interval. The Remove method removes a NonworkingPattern objects. The for each statement is supported by NonworkingPatterns collection, so the NonworkingPatterns collection can be enumerated using a sample like *for each nw in NonworkingPatterns*

property NonworkingPatterns.Item (ID as Variant) as NonworkingPattern

Returns a specific NonworkingPattern of the NonworkingPatterns collection, giving its identifier.

Туре	Description
ID as Variant	A long expression that defines the identifier of the NonworkingPattern object.
<u>NonworkingPattern</u>	A NonworkingPattern object associated with the giving identifier, or empty/nothing/NULL, if no object is associated with the specified identifier.

The Item property accesses a NonworkingPattern object based on its identifier. The Count property counts the number of NonworkingPattern objects in the NonworkingPatterns collection. The ID property indicates the identifier of the NonworkingPattern object. The IDNonworkingPattern property of the NonworkingTime object indicates the identifier of the NonworkingPattern object to be displayed on the non-working interval. The Remove method removes a NonworkingPattern objects. The **for each statement** is supported by NonworkingPatterns collection, so the NonworkingPatterns collection can be enumerated using a sample like *for each nw in NonworkingPatterns*

method NonworkingPatterns.Remove (ID as Variant)

Removes a specific member from the NonworkingPatterns collection.

Туре	Description
ID as Variant	A long expression that specifies the identifier of the NonworkingPattern object to be removed.

The <u>Remove</u> method removes a NonworkingPattern object based on its identifier. The <u>Clear</u> method removes all defined <u>NonworkingPattern</u> objects. The <u>Count</u> property counts the number of NonworkingPattern objects in the NonworkingPatterns collection. The <u>Item</u> property accesses a NonworkingPattern object based on its identifier. The <u>ID</u> property indicates the identifier of the NonworkingPattern object. The <u>IDNonworkingPattern</u> property of the <u>NonworkingTime</u> object indicates the identifier of the NonworkingPattern object to be displayed on the non-working interval.

NonworkingTime object

The NonworkingTime object defines the non-working time-interval. Each NonworkingTime object can associate a <u>NonworkingPattern</u> object to define the colors/pattern to display the non-working time-interval. The <u>NonworkingPattern</u> object holds the colors/pattern to display a non-working time-interval. The <u>ID</u> property of NonworkingPattern defines the identifier of the pattern/color. The <u>IDNonworkingPattern</u> property of the NonworkingTime object indicates the identifier of the <u>NonworkingPattern</u> object to be displayed on the non-working interval.

The NonworkingPattern object supports the following properties and methods:

Name	Description
EndTime	Indicates the end time of the non-working zone.
Expression	Indicates the expression that defines the non-working zone.
<u>GroupID</u>	Specifies the ID of Group objects to apply the nonworking- time.
<u>Handle</u>	Gets handle of the NonworkingTime object.
IDNonworkingPattern	Indicates the pattern to show the non-working zone.
Index	Gets the index of the NonworkingTime object in its collection.
<u>IsValid</u>	Specifies the giving expression is valid, so the non-working zone is applicable.
NonworkingPattern	Indicates the current pattern to display the non-working zone.
<u>StartTime</u>	Indicates the start time of the non-working zone.
property NonworkingTime.EndTime as String

Indicates the end time of the non-working zone.

Туре	Description
String	A String expression that defines the time to end of the non-working time-interval. For instance, the "08:00" indicates the an 8 AM, while the "16:15" indicates the 8:15 PM

The EndTime property is initialized with the EndTime parameter of the <u>Add</u> method. The EndTime property can be used to programmatically extends the non-working time-interval. The <u>StartTime</u>/EndTime properties defines the interval of time the non-working part is shown. The <u>IsValid</u> property indicates whether the non-working time-interval is valid, and so if it is visible or hidden. The <u>Expression</u> property defines the expression that specifies the days where the non-working time-interval is displayed. The <u>IDNonworkingPattern</u> property specifies the identifier of the NonworkingPattern object associated with the current non-working time-interval.

property NonworkingTime.Expression as String

Indicates the expression that defines the non-working zone.

Туре	Description
String	A String expression that defines the formula to determine when the non-working time-interval is displayed.

The Expression property is initialized with the Expression parameter of the Add method. The Expression property supports the **value** keyword and predefined operators and functions like explained bellow. *The <u>IsValid</u> property indicates whether the non-working time-interval is valid, and so if it is visible or hidden.* The Expression property defines the expression that specifies the days where the non-working time-interval is displayed. The <u>IDNonworkingPattern</u> property specifies the identifier of the NonworkingPattern object associated with the current non-working time-interval.

Here's a few samples of expressions:

- "0", the non-working time-interval is hidden
- "1", the non-working time-interval is shown every day
- "weekday(value) = 1", the non-working time-interval is shown every Monday
- "weekday(value) in (1,2) and month(value) = 6", the non-working time-interval is shown every Monday and Tuesday, on June only.
- "value in (#6/8/2012#,#6/11/2012#,#6/20/2012#)", the non-working time-interval is shown on 6/8/2012, 6/11/2012 and 6/20/2012
- "value >= #6/1/2012# and ((value #6/1/2012#) mod 5 = 0)", the schedule view displays the non-working time-interval starting from 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 schedule view displays the non-working time-interval starting from 6/1/2012, and shows up 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.

The Expression 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#, #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:

- \circ 0 empty (not initialized)
- **1 null**

•

- 2 short
- 3 long
- 4 float
- 5 double
- \circ 6 currency
- 7 date
- \circ 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
- Itrim (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)

property NonworkingTime.GroupID as Variant

Specifies the ID of Group objects to apply the nonworking-time.

Туре	Description
Variant	Could be a numeric expression, a string of numbers separated by comma, or a one-dimensional safe array of values that indicates the ID of the Group objects to apply the current nonworking time object.

By default, the GroupID is empty, which indicates that the non-working time is applied to all groups. The GroupID property specifies the list of groups that displays the current non-working time. Use the GroupID property to specify a different non-working time for different groups. The ID property of the Group object indicates the identifier of the group to be displayed on the scheduler. Use the Add method to add new groups to the control.

property NonworkingTime.Handle as Long

Gets handle of the NonworkingTime object.

Туре	Description
Long	A Long expression that specifies the handle of the NonworkingTime object.

Each NonworkingTime object has assigned an unique handle, which is allocated at creating time. The Handle property can not be changed at runtime. The <u>Index</u> property may be used to identify a NonworkingTime object based on its index in the NonworkingTimes collection. The <u>ShowNonworkingTime</u> property shows or hides the defined non-working intervals. The <u>NonworkingTimeFromPoint</u> property gets the non-working object from the cursor. The <u>Item</u> property can access a NonworkingTime object based on its handle, reference or index.

property NonworkingTime.IDNonworkingPattern as Long

Indicates the pattern to show the non-working zone.

l

Туре	Description
Long	A Long expression that defines the identifier of the <u>NonworkingPattern</u> object to be displayed on the non- working time-interval.

The IDNonworkingPattern property indicates the identifier of the NonworkingPattern object to be displayed on the non-working interval. The ID property of NonworkingPattern defines the identifier of the pattern/color. The <u>NonworkingPattern</u> property of the NonworkingTime object gives access directly to the associated <u>NonworkingPattern</u> object. You can still access the <u>NonworkingPattern</u> object using the <u>Item</u> property based on the identifier. You can use the IDNonworkingPattern property to programmatically change the visual appearance of the non-working time-interval.

property NonworkingTime.Index as Long

l

Gets the index of the NonworkingTime object in its collection.

Туре	Description
Long	A Long expression that defines the index of the NonworkingTime object in its collection

The Index property of the NonworkingTime object indicates the INDEX of the object within its collection, *NOT the identifier that should be used on the <u>IDNonworkingPattern</u> property. The <u>Item</u> property can access a NonworkingTime object based on its handle, reference or index. The <u>Count</u> property indicates the number of NonworkingTime object in the NonworkingTimes collection. The for each statement is supported by the NonworkingTimes collection, so the NonworkingTimes collection can be enumerated using a sample like <i>for each nw in NonworkingTimes*

property NonworkingTime.lsValid as Boolean

Specifies the giving expression is valid, so the non-working zone is applicable.

Туре	Description
Boolean	A Boolean expression that specifies whether the Expression of the non-working object is valid (syntactically correct)

The IsValid property gets True, if the Expression property is syntactically correct. If the Expression property is not valid or is syntactically incorrect, the IsValid property returns False, The non-working time interval is visible ONLY, if the IsValid property is True. In other words, you can use the Expression to define whether a non-working time-interval is visible or hidden. For instance, you can add add an "A" character at the start of the Expression, and so the expression is not valid, and so no shown. NExt, remove the "A" character in front, and the non-working time-interval will be visible again.

property NonworkingTime.NonworkingPattern as NonworkingPattern

Indicates the current pattern to display the non-working zone.

Туре	Description
NonworkingPattern	A NonworkingPattern object associated with the current non-working time-interval

The NonworkingPattern property of the NonworkingTime object gives access directly to the associated <u>NonworkingPattern</u> object. You can still access the <u>NonworkingPattern</u> object using the <u>Item</u> property based on the identifier. The <u>IDNonworkingPattern</u> property indicates the identifier of the NonworkingPattern object to be displayed on the non-working interval. The <u>ID</u> property of NonworkingPattern defines the identifier of the pattern/color. You can use the <u>IDNonworkingPattern</u> property to programmatically change the visual appearance of the non-working time-interval.

property NonworkingTime.StartTime as String

Indicates the start time of the non-working zone.

Туре	Description
String	A String expression that defines the time to start of the non-working time-interval. For instance, the "08:00" indicates the an 8 AM, while the "16:15" indicates the 8:15 PM

The StartTime property is initialized with the StartTime parameter of the Add method. The StartTime property can be used to programmatically extends the non-working time-interval. The StartTime/EndTime properties defines the interval of time the non-working part is shown. The IsValid property indicates whether the non-working time-interval is valid, and so if it is visible or hidden. The Expression property defines the expression that specifies the days where the non-working time-interval is displayed. The IDNonworkingPattern property specifies the identifier of the NonworkingPattern object associated with the current non-working time-interval.

NonworkingTimes object

The NonworkingTimes object holds a collection of <u>NonworkingTime</u> objects. The NonworkingTime object indicates a time interval to be shown as non-working. Each NonworkingTime object can associate a <u>NonworkingPattern</u> object that specifies the colors and the pattern to show the non-working zone. The NonworkingTimes collection is accessible through the <u>NonworkingTimes</u> property of the control. The <u>NonworkingPatterns</u> collection is accessible through the <u>NonworkingPatterns</u> property of the control. The <u>NonworkingDays</u> property of the calendar defines the days to be non-working in the calendar. The <u>ShowNonworkingTime</u> property shows or hides the defined non-working intervals.

The NonworkingTime's advantages are:

- highlight the interval of time as non-working with a different patterns, colors.
- any/all day can display different intervals of time as non-working
- you can specify the non-working interval using an expression, that defines the days where the non-working interval is shown.

The following screen shot shows different days with different non-working area:



The NonworkingTimes object supports the following properties and methods:

Name	Description
Add	Adds a NonworkingTime object to the collection and returns a reference to the newly created object.
<u>Clear</u>	Removes all objects in a collection.
<u>Count</u>	Returns the number of objects in a collection.
<u>ltem</u>	Returns a specific NonworkingTime of the NonworkingTimes collection, giving its handle or index.
<u>Remove</u>	Removes a specific member from the NonworkingTimes collection, giving its index, handle or reference.

method NonworkingTimes.Add (Expression as String, StartTime as String, EndTime as String, IDNonworkingPattern as Long)

Adds a NonworkingTime object to the collection and returns a reference to the newly created object.

Туре	Description
Expression as String	A String expression that defines the days where the non- working time-interval is shown. For instance, the "weekday(value) = 1" indicates that the non-working time- interval should be displayed on Mondays . The <u>Expression</u> property supports value keyword among with a lot of operators and predefined functions.
StartTime as String	A String expression that specifies the time to start the non- working time-interval. For instance, the "08:00" indicates the an 8 AM, while the "16:15" indicates the 8:15 PM
EndTime as String	A String expression that specifies the time to end the non- working time-interval. For instance, the "08:00" indicates the an 8 AM, while the "16:15" indicates the 8:15 PM
IDNonworkingPattern as Long	A Long expression that specifies the identifier of the <u>NonworkingPattern</u> object that will be associated with the newly created non-working time-interval
Return	Description
Nonworking Time	A NonworkingTime object being created.

The Add method of the NonworkingTimes objects adds a new non-working time interval. The <u>Expression</u> property indicates the expression that defines the dates to include the specified non-working interval. *The <u>IsValid</u> property indicates whether the non-working expression is valid, and so, if it is visible or hidden.* The <u>StartTime/EndTime</u> property defines the time to start/end the non-working time-zone. The <u>ShowNonworkingTime</u> property shows or hides the defined non-working intervals. The <u>NonworkingTimeFromPoint</u> property gets the non-working object from the cursor.

The NonworkingTimes object holds a collection of <u>NonworkingTime</u> objects. The NonworkingTime object indicates a time interval to be shown as non-working. Each NonworkingTime object can associate a <u>NonworkingPattern</u> object that specifies the colors and the pattern to show the non-working zone. The <u>NonworkingPatterns</u> collection is accessible through the <u>NonworkingPatterns</u> property of the control. The <u>NonworkingDays</u> property of the calendar defines the days to be non-working in the calendar.

The NonworkingTime's advantages are:

- highlight the interval of time as non-working with a different patterns, colors.
- any/all day can display different intervals of time as non-working
- you can specify the non-working interval using an expression, that defines the days where the non-working interval is shown.

method NonworkingTimes.Clear ()

Removes all objects in a collection.

Туре

Description

The Clear method removes all elements in the NonworkingTimes collection, or clears all non-working time-intervals. The <u>ShowNonworkingTime</u> property shows or hides the added non-working time-intervals. The <u>Remove</u> method removes a specific non-working time-zone. The <u>NonworkingTimeFromPoint</u> property gets the non-working object from the cursor. The <u>Add</u> method of the NonworkingTimes objects adds a new non-working time interval.

property NonworkingTimes.Count as Long

Returns the number of objects in a collection.

Туре	Description
Long	A long indicates the number of NonworkingTime object in the NonworkingTimes collection

The Count property indicates the number of NonworkingTime object in the NonworkingTimes collection. The <u>Item</u> property can access a NonworkingTime object based on its handle, reference or index. The **for each statement** is supported by the NonworkingTimes collection, so the NonworkingTimes collection can be enumerated using a sample like *for each nw in NonworkingTimes*

property NonworkingTimes.Item (Handle as Variant) as NonworkingTime

Returns a specific NonworkingTime of the NonworkingTimes collection, giving its handle or index.

Туре	Description
Handle as Variant	A Long expression that specifies the index or the handle of the NonworkingTime object being accessed.
<u>NonworkingTime</u>	A NonworkingTime object being requested, or empty/nothing/NULL if no object found with associated index of handle.

The Item property can access a NonworkingTime object based on its handle, reference or index. The <u>Count</u> property indicates the number of NonworkingTime object in the NonworkingTimes collection. The **for each statement** is supported by the NonworkingTimes collection, so the NonworkingTimes collection can be enumerated using a sample like *for each nw in NonworkingTimes*

method NonworkingTimes.Remove (Handle as Variant)

Removes a specific member from the NonworkingTimes collection, giving its index, handle or reference.

Туре	Description
Handle as Variant	A long expression that specifies the handle or the index of the NonworkingTime object to be removed. You can pass also the NonworkingTime object reference to be removed.

The Remove method removes a specific non-working time-zone. The <u>Clear</u> method removes all elements in the NonworkingTimes collection, or clears all non-working time-intervals. The <u>ShowNonworkingTime</u> property shows or hides the added non-working time-intervals. The <u>NonworkingTimeFromPoint</u> property gets the non-working object from the cursor. The <u>Add</u> method of the NonworkingTimes objects adds a new non-working time interval.

Pattern object

The Pattern object can be used to apply a pattern and a frame with different colors on an UI element. For instance, the <u>Calendar</u>.<u>HighlightEvent</u>.<u>Pattern</u> property indicates the pattern to be applied on the calendar panel for dates with events. The Event.<u>BodyPattern</u> property indicates the pattern to be shown on the event's body. The Pattern object supports the following properties:

Name	Description
<u>Color</u>	Specifies the pattern color.
FrameColor	Specifies the pattern's frame color.
Туре	Retrieves or sets a value that indicates the pattern to fill the element.

property Pattern.Color as Color

Specifies the pattern color.

Туре	Description
Color	A Color expression that specifies the color to show the pattern.

By default, the Color property is 0 (black). The Color property indicates the color to display the pattern. The <u>Type</u> property indicates the type of the pattern to be shown. The <u>FrameColor</u> property indicates the color to show the frame, if the exPatternFrame flag is included in the Type property.

property Pattern.FrameColor as Color

Specifies the pattern's frame color.

Туре	Description
Color	A Color expression that specifies the color to show the frame.

By default, the FrameColor property is 0 (black). The FrameColor property indicates the color to show the frame, if the exPatternFrame flag is included in the <u>Type</u> property. The <u>Type</u> property indicates the type of the pattern to be shown. The <u>Color</u> property indicates the color to display the pattern.

property Pattern.Type as PatternEnum

Retrieves or sets a value that indicates the pattern to fill the element.

Туре	Description
PatternEnum	A PatternEnum expression that specifies the type of the pattern to fill the element.

By default, the Type property is exPatternEmpty which indicates that no pattern is shown. The Type property indicates the pattern to display on the element. The <u>Color</u> property indicates the color to display the pattern. The <u>FrameColor</u> property indicates the color to show the frame, if the exPatternFrame flag is included in the Type property.

Schedule 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: {9B09E13D-7A88-4299-9DBE-383380435377}. The object's program identifier is: "Exontrol.Schedule". The /COM object module is: "ExSchedule.dll"

The <u>Calendar</u> property returns the calendar panel of the schedule view. Use the <u>OnResizeControl</u> property on exCalendarAutoHide, to auto hide the calendar's panel.

The following screen shot shows the control with a picture on its background (tiled):



The Schedule object supports the following properties and methods:

Name	Description
<u>AllowAllDayEventScroll</u>	Gets or sets a value that specifies whether the all-day event header supports scrolling.
AllowCreateAllDayEvent	Specifies whether the user can create all day events.
<u>AllowCreateEvent</u>	Specifies keys combination that allows the user to create events in the schedule view.
<u>AllowEditEvent</u>	Specifies keys combination that allows the user to edit the event's caption in the schedule view.
<u>AllowExchangePanels</u>	Exchanges the panels when the user clicks the giving keys combination and drag the panel to a new position.
<u>AllowMoveEvent</u>	Specifies the combination of keys that allows the user to move the event.
<u>AllowMoveEventToOtherGro</u>	Specifies if the event can ve moved from a group to up another when dragging.

<u>AllowMoveGroup</u>	Specifies the combination of keys that allows the user to move the group.
<u>AllowMoveMarkTime</u>	Specifies the combination of keys that allows the user to move a mark time.
<u>AllowMoveSchedule</u>	Specifies the combination of keys that allows the user to move the schedule view.
<u>AllowMoveTimeScale</u>	Specifies the combination of keys to move the control's time scale.
<u>AllowMultiDaysEvent</u>	Specifies whether the user can create events that may start on a day and ends on other.
<u>AllowRefineMoveKey</u>	Specifies the combination of keys to refine the start and end of events when creating, moving or resizing the events.
AllowResizeEvent	Specifies the combination of keys that allows the user to resize the event.
<u>AllowResizeGroup</u>	Specifies the combination of keys that allows the user to resize the group.
<u>AllowResizeSchedule</u>	Specifies the combination of keys that allows the user to resize the schedule view.
<u>AllowResizeTimeScale</u>	Specifies the combination of keys that allows the user to resize the time scale.
<u>AllowSelectCreateEvent</u>	Specifies whether the newly created event gets selected or highlighted
<u>AllowSelectEvent</u>	Specifies the combination of keys that allows the user to select events in the schedule panel.
<u>AllowSelectEventRect</u>	Specifies the combination of keys that allows the user to select events in the schedule panel, by dragging a rectangle.
AllowToggleSchedule	Toggles the schedule view when user double clicks it.
<u>AllowToggleSelectKey</u>	Specifies the combination of keys to select multiple not- contiguously events.
<u>AllowUndoRedo</u>	Enables or disables the Undo/Redo feature.
<u>AllowUpdateAllDayFlag</u>	Indicates if the All-Day flag for the events being moved using drop and drop are updated once the user drops the selection.
<u>AllowUpdateDisableZone</u>	Indicates whether the user can updates the events in the disabled part of the schedule.

AnchorFromPoint	Retrieves the identifier of the anchor from point.
Appearance	Retrieves or sets the control's appearance.
<u>ApplyGroupingColors</u>	Specifies whether the schedule view shows the events using the colors of owner groups.
AttachTemplate	Attaches a script to the current object, including the events, from a string, file, a safe array of bytes.
BackColor	Specifies the control's background color.
Background	Returns or sets a value that indicates the background color for parts in the control.
<u>BeginUpdate</u>	Maintains performance when items are added to the control one at a time. This method prevents the control from painting until the EndUpdate method is called.
BodyEventBackColor	Specifies the default visual appearance of the events.
BodyEventForeColor	Specifies the default foreground color of the events.
BorderDateStyle	Specifies the style to display the border for the dates.
BorderGroupStyle	Specifies the style to display the border between groups within the date.
<u>BorderHeight</u>	Sets or retrieves a value that indicates the border height of the control.
BorderMonthStyle	Specifies the style to display the border for the months.
BorderSelStyle	Specifies the style to display the border for selected dates.
BorderTimeScaleStyle	Specifies the style to display the border for time scales.
<u>BorderWidth</u>	Sets or retrieves a value that indicates the border width of the control.
<u>Calendar</u>	Gets the schedule's calendar object.
<u>CanRedo</u>	Retrieves a value that indicates whether the surface can perform a Redo operation.
<u>CanUndo</u>	Retrieves a value that indicates whether the surface can perform an Undo operation.
<u>ClearAll</u>	Clears all control's collections, including the events.
<u>ClipToSel</u>	Specifies whether the schedule view displays the selection only.
Copy	Copies the control's content to the clipboard, in the EMF format.

<u>CopyTo</u>	Exports the control's view to an EMF file.
<u>CreateEventLabel</u>	Specifies the label to be shown while creating events.
<u>CreateEventLabelAlign</u>	Specifies the alignment of the label to be shown while creating events.
<u>DataField</u>	Automatically updates / synchronizes the known property of the event with the associated data field and reverse.
<u>DataSource</u>	Retrieves or sets a value that indicates the data source for object.
<u>DateEvents</u>	Returns a safe array of Event objects in a giving date.
DateTimeFromPoint	Retrieves the date/time from the cursor, in the schedule panel.
<u>DayEndTime</u>	Indicates the day end time.
DayStartTime	Indicates the day start time.
<u>DayViewHeight</u>	Indicates the height of the day's view in the schedule panel.
<u>DayViewOffsetX</u>	Indicates the horizontal scroll position of the schedule's view.
<u>DayViewOffsetY</u>	Indicates the vertical scroll position of the schedule's view.
<u>DayViewWidth</u>	Indicates the width of the day's view in the schedule panel.
DefaultEventLongLabel	Indicates the default long label for events.
DefaultEventPadding	Returns or sets a value that indicates the padding of the events in the control.
DefaultEventShortLabel	Indicates the default short label for events.
DefaultEventTooltip	Indicates the default tooltip for events.
Description	Changes descriptions for control objects.
DisplayGroupingButton	Gets or sets a value that indicates whether the grouping button is displayed in the date header.
EditContextMenultems	Specifies the control's context menu, while editing the event.
Enabled	Enables or disables the control.
EndBlockUndoRedo	Ends recording the UI operations and adds the undo/redo operations as a block, so they all can be restored at once, if Undo method is performed.
	Resumes painting the control after painting is suspended

EndUpdate	by the BeginUpdate method.
EnsureVisible	Ensures that the specified date fits the client area of the schedule view.
EventFromPoint	Gets the Event object from the cursor, in the schedule panel.
<u>EventParam</u>	Retrieves or sets a value that indicates the current's event parameter.
<u>Events</u>	Gets the Events collection of the scheduler.
<u>EventsFont</u>	Retrieves or sets the font to display the events in the schedule view.
EventsTransparent	Specifies the percent of transparency to show the events in the schedule panel.
ExecuteTemplate	Executes a template and returns the result.
FitSelToView	Fits the selected dates to the current view.
<u>Font</u>	Retrieves or sets the control's font.
<u>ForeColor</u>	Specifies the control's foreground color.
<u>FormatAnchor</u>	Specifies the visual effect for anchor elements in HTML captions.
<u>GroupFromPoint</u>	Retrieves the Group object from the cursor, in the schedule panel.
<u>GroupHeaderFromPoint</u>	Retrieves the Group's header from the cursor, in the schedule panel.
<u>GroupHighlightEvent</u>	Highlights the date in the schedule panel using the HighlightEvent property of each Group found on day's events.
<u>Groups</u>	Retrieves the Groups collection of the scheduler.
GroupUndoRedoActions	Groups the next to current Undo/Redo Actions in a single block.
HeaderAllDayEventHeight	Specifies the height of the All-Day events being displayed on the control's All-Day header.
HeaderDayHeight	Indicates the height of the day's header.
HeaderDayLongLabel	Specifies the long label for header days.
HeaderDayShortLabel	Specifies the short label for header days.
HeaderGroupHeight	Indicates the height of the group's header.

<u>HighlightDate</u>	Highlights the specified date.
HTMLPicture	Adds or replaces a picture in HTML captions.
<u>hWnd</u>	Retrieves the control's window handle.
Images	Sets at runtime the control's image list. The Handle should be a handle to an Images List Control.
ImageSize	Retrieves or sets the size of icons the control displays
Layout	Saves or loads the control's layout, such selection in calendar/schedule panels, scroll bars, grouping, and so on.
LoadXML	Loads an XML document from the specified location, using MSXML parser.
MarkTimeFromPoint	Gets the MarkTime object from the cursor, in the schedule panel.
<u>MarkTimes</u>	Gets the MarkTimes collection of the scheduler.
MarkZoneFromPoint	Retrieves the MarkZone from the cursor, in the schedule panel.
<u>MarkZones</u>	Retrieves the MarkZones collection of the scheduler.
NonworkingPatterns	Retrieves the NonworkingPatterns collection of the scheduler.
NonworkingTimeFromPoint	Retrieves the NonworkingTime from the cursor, in the schedule panel.
<u>NonworkingTimes</u>	Retrieves the NonworkingTimes collection of the scheduler, to specify different non-working time for different days.
<u>OLEDrag</u>	Causes a component to initiate an OLE drag/drop operation.
<u>OLEDropMode</u>	Returns or sets how a target component handles drop operations
OnResizeControl	Specifies which panel is resized when the control is resized.
PaneMinWidth	Specifies the minimum width for the left or right panel.
PaneWidth	Specifies the width for the left or right panel.
<u>Picture</u>	Retrieves or sets a graphic to be displayed in the control.
<u>PictureDisplay</u>	Retrieves or sets a value that indicates the way how the graphic is displayed on the control's background

PictureFromPoint	Retrieves the identifier of the picture from the point (Event.Pictures or Event.ExtraPictures).
Pictures	Gets the Pictures collection of the scheduler.
Redo	Redoes the next action in the surface's Redo queue.
RedoListAction	Lists the Redo actions that can be performed on the surface.
RedoRemoveAction	Removes the first redo actions that can be performed on the surface.
<u>Refresh</u>	Refreshes the control.
RemoveSelection	Removes the selected events.
<u>ReplaceIcon</u>	Adds a new icon, replaces an icon or clears the control's image list.
<u>SaveXML</u>	Saves the control's content as XML document to the specified location, using the MSXML parser.
<u>ScrollBars</u>	Returns or sets a value that determines whether the control has horizontal and/or vertical scroll bars.
ScrollButtonHeight	Specifies the height of the button in the vertical scrollbar.
ScrollButtonWidth	Specifies the width of the button in the horizontal scrollbar.
<u>ScrollFont</u>	Retrieves or sets the scrollbar's font.
<u>ScrollHeight</u>	Specifies the height of the horizontal scrollbar.
ScrollOrderParts	Specifies the order of the buttons in the scroll bar.
ScrollPartCaption	Specifies the caption being displayed on the specified scroll part.
ScrollPartCaptionAlignment	Specifies the alignment of the caption in the part of the scroll bar.
ScrollPartEnable	Indicates whether the specified scroll part is enabled or disabled.
<u>ScrollPartVisible</u>	Indicates whether the specified scroll part is visible or hidden.
<u>ScrollThumbSize</u>	Specifies the size of the thumb in the scrollbar.
<u>ScrollToolTip</u>	Specifies the tooltip being shown when the user moves the scroll box.
<u>ScrollWidth</u>	Specifies the width of the vertical scrollbar.
	Indicates the number of events being selected in the

<u>SelCount</u>	schedule panel.
SelectAll	Selects all events in the control.
<u>SelectEventColor</u>	Indicates the color to show the selected events.
<u>SelectEventStyle</u>	Specifies the style to display the selected event.
SelectEventTextColor	Indicates the color to show the text for selected events.
<u>Selection</u>	Returns or sets a safe array of selected events in the schedule panel.
<u>SelEvent</u>	Gets the event being selected giving its index in the selection.
ShowAllDayHeader	Specifies whether the control shows or hides the header for All-Day events.
ShowEventLabels	Indicates whether the Label or ExtraLabel of the events are being shown or hidden.
ShowEventPictures	Indicates whether the Pictures or ExtraPictures of the events are being shown or hidden.
ShowEvents	Indicates the type of the events which schedule displays.
<u>ShowGroupingEvents</u>	Specifies whether the schedule view shows grouped events.
ShowHighlightDate	Returns or sets a value that indicates whether the control shows the highlighted dates.
<u>ShowHighlightEvent</u>	Returns or sets a value that indicates whether the schedule panel highlights days that contain events.
<u>ShowImageList</u>	Specifies whether the control's image list window is visible or hidden.
ShowMarkTime	Indicates whether the schedule shows the mark times.
ShowMarkZone	Indicates how the schedule panel shows the mark zones.
ShowNonworkingTime	Returns or sets a value that indicates whether the schedule panel displays nonworking time.
ShowSelectEvent	Specifies whether the selected events are highlighted.
ShowStatusEvent	Gets or sets a value that specifies whether the event's status is visible or hidden.
ShowTimeScale	Specifies whether the control's time scale is shown on the schedule panel.
Show Tool Tip	Shows the specified tooltip at given position.

<u>ShowViewCompact</u>	Indicates whether the schedule view is compact, so the first day of the month starts right after the last day of the previously month, or start to a new row.
<u>SingleGroupingView</u>	Indicates whether the schedule shows single or multiple groups of events at once.
StartBlockUndoRedo	Starts recording the UI operations as a block of undo/redo operations.
StatusEventColor	Indicates the default visual appearance for the event's status.
StatusEventSize	Indicates the size of the event's status.
<u>Synchronize</u>	Synchronizes the control' events with the records, while the control is bounded to a recordset, using the DataSource property.
Template	Specifies the control's template.
<u>TemplateDef</u>	Defines inside variables for the next Template/ExecuteTemplate call.
<u>TemplatePut</u>	Defines inside variables for the next Template/ExecuteTemplate call.
TimeFromPoint	Retrieves the time from the cursor, in the schedule panel.
<u>TimeScaleFont</u>	Retrieves or sets the font to display the time scales in the schedule view.
TimeScaleFromPoint	Retrieves the TimeScale object from the cursor, in the schedule panel.
<u>TimeScales</u>	Gets the schedule's time scales collection.
<u>ToolTipDelay</u>	Specifies the time in ms that passes before the ToolTip appears.
<u>ToolTipFont</u>	Retrieves or sets the tooltip's font.
<u>ToolTipPopDelay</u>	Specifies the period in ms of time the ToolTip remains visible if the mouse pointer is stationary within a control.
<u>ToolTipWidth</u>	Specifies a value that indicates the width of the tooltip window, in pixels.
Undo	Performs the last Undo operation.
<u>UndoListAction</u>	Lists the Undo actions that can be performed on the surface.
<u>UndoRedoQueueLength</u>	Gets or sets the maximum number of Undo/Redo actions that may be stored to the surface's queue.
<u>UndoRemoveAction</u>	Removes the last undo actions that can be performed on the surface.
-------------------------------	--
<u>UpdateEventsLabel</u>	Specifies the label to be shown while moving or resizing the events.
<u>UpdateEventsLabelAlign</u>	Specifies the alignment of the label to be shown while moving or resizing events.
<u>UseVisualTheme</u>	Specifies whether the control uses the current visual theme to display certain UI parts.
Version	Retrieves the control's version.
VerticalScrollWheel	Indicates the distance to scroll using the mouse wheel.
<u>VisualAppearance</u>	Retrieves the control's appearance.
<u>VisualDesign</u>	Invokes the control's VisualAppearance designer.

property Schedule.AllowAllDayEventScroll as AllDayEventScrollEnum

Gets or sets a value that specifies whether the all-day event header supports scrolling.

Туре

Description

<u>AllDayEventScrollEnum</u>

An <u>AllDayEventScrollEnum</u> expression that specifies whether the all-day header supports scrolling.

By default, the AllowAllDayEventScroll property is exAllDayEventWheelScroll. The AllowAllDayEventScroll property gets or sets a value that specifies whether the all-day event header supports scrolling. Use the <u>ShowAllDayHeader</u> property to show the schedule's All-Day header so all All-Day evens are shown on this header. The <u>AllowCreateAllDayEvent</u> property has effect only when the schedule displays no time scale. *At runtime, the user can create all-day events, if the schedule view displays no time scale. When the schedule view shows several dates (or it is resized to the minimum), so no time scale is available, the user can create all-day events by clicking a date and dragging the mouse to the day where the event should end.* The AddEvent event is fired once the user creates a new event. The <u>AllDayEvent</u> property indicates an all-day event. The <u>AllowSelectCreateEvent</u> property specifies whether the newly created event gets selected or highlighted.



property Schedule.AllowCreateAllDayEvent as Boolean

Specifies whether the user can create all day events.

Туре	Description
Boolean	A Boolean expression that specifies whether the user can create all-day events at runtime.

By default, the AllowCreateAllDayEvent property is True. The AllowCreateAllDayEvent property has effect only when the schedule displays no time scale. *At runtime, the user can create all-day events, if the schedule view displays no time scale. When the schedule view shows several dates (or it is resized to the minimum), so no time scale is available, the user can create all-day events by clicking a date and dragging the mouse to the day where the event should end. Use the ShowAllDayHeader property to show the schedule's All-Day header so all All-Day evens are shown on this header. The AddEvent event is fired once the user creates a new event. The UpdateEvent event is fired once the margins of the event is being changed. The Start and End properties of the Event indicates the margins of the event. The AllDayEvent property indicates an all-day event. The AllowAllDayEventScroll property gets or sets a value that specifies whether the all-day event header supports scrolling. The AllowSelectCreateEvent property specifies whether the newly created event gets selected or highlighted.*

The <u>AllowCreateEvent</u> property specifies the keys combination that allows the user to create events in the schedule view. The <u>AllowMultiDaysEvent</u> property indicates whether the user can create events that may start on a day and ends on other. The <u>CreateEventLabel</u> property indicates the HTML format to be shown on the label when the user creates a new event. The <u>CreateEventLabelAlign</u> property aligns the label being shown when the user creates a new event.

The following screen shot shows the control with no time scale (the schedule view has been resized using the MIDDLE mouse button):

Jan 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	Jan 31				
/			Feb 1	2	3	4

Newly created events have the <u>AllDayEvent</u> property on True, as the time scale is not available

The following screen shot shows the control with time scale (the schedule view has been resized using the MIDDLE mouse button):

	Sunday	Monday	Tuesday	Wednesday
08:00 AM		1/2/2012		
:30				
09:00 AM :30				
10:00 AM :30				
11:00 AM		21 0 - 10 - 10 - 10		
12:00 PM				••
01:00 PM		19-1 - S ¹⁻¹		
02:00 PM :30				
03:00 PM :30				
	Sunday	Monday	Tuesday	Wednesday
08:00 AM			1/10/2012	
09:00 AM				
:30				
10:00 AM :30				

Newly created events have the <u>AllDayEvent</u> property on False, as the time scale is shown

property Schedule.AllowCreateEvent as AllowKeysEnum

Specifies keys combination that allows the user to create events in the schedule view.

Туре	Description
<u>AllowKeysEnum</u>	An AllowKeysEnum expression that specifies the keys combination so user can create new events at runtime.

By default, the AllowCreateEvent property is exLeftClick, which indicates that the user can press the left mouse button to start creating a new event. The AllowCreateEvent property on exDisallow indicates that user can not creates new events. The AllowCreateEvent property indicates the combination of the keys to let user creates new events. The AddEvent event is fired once the user creates a new event. The AllowCreateAllDayEvent property specifies whether the user can create all day events, at runtime. The AllowMultiDaysEvent property indicates whether the user can create events that may start on a day and ends on other. The AllowRefineMoveKey property specifies whether the margins of the events being updated are aligned to minor/major rulers of the control's time scale.

The <u>CreateEventLabel</u> property indicates the HTML format to be shown on the label when the user creates a new event. The <u>CreateEventLabelAlign</u> property aligns the label being shown when the user creates a new event. The <u>AllowMoveEvent</u> property indicates the combination of the keys to let user moves the events. The <u>AllowResizeEvent</u> property indicates the combination of the keys to let user resizes the events.

The <u>Background</u>(exScheduleCreateEventBackColor) and <u>Background</u>(exScheduleCreateEventForeColor) specifies the visual appearance of the event being created.

property Schedule.AllowEditEvent as AllowKeysEnum

Specifies keys combination that allows the user to edit the event's caption in the schedule view.

Туре	Description
<u>AllowKeysEnum</u>	An AllowKeysEnum expression that specifies the keys to start inline editing.

By default, the AllowEditEvent property is exLeftClick + exDblClick, which means that the user can double click an event so the inline editing starts. The AllowEditEvent property specifies the combination of keys that the user can use so the event gets inline editing. The AllowEditEvent property on exDisallow, specifies that the user can not use inline editing. The Editable property indicates the property of the event to be edited when user double clicks the event. The LayoutStartChanging(exScheduleEditEvent) event notifies your application once the inline editing starts. The LayoutEndChanging(exScheduleEditEvent) event notifies your application once the inline editing starts. You can use the EventFromPoint(-1,-1) method during the LayoutStartChanging(exScheduleEditEvent) to store the event from the cursor to a global member, and when LayoutEndChanging(exScheduleEditEvent) occurs, you can use the previously stored member to identify the event being edited.

You can specify an event being not editable, using the <u>Editable</u> property on exNoEditable, or you can set the AllowEditEvent property on exDisallow, to prevent editing any of the schedule's events.

The AllowEditEvent property uses the same keys combination as <u>AllowToggleSchedule</u> property, so if you double click an event, the inline editing is performed, else if clicking outside of the event the toggling the schedule view is performed.

property Schedule.AllowExchangePanels as AllowKeysEnum

Exchanges the panels when the user clicks the giving keys combination and drag the panel to a new position.

Туре	Description
<u>AllowKeysEnum</u>	An AllowKeysEnum expression that specifies the keys the user can exchange the position of the calendar/schedule view.

By default, the AllowExchangePanels property is exLeftClick + exSHIFTKey + exCTRLKey. The AllowExchangePanels property specifies whether the user can drag and drop a panel (calendar panel or schedule panel) to a new position/alignment. For instance, the AllowExchangePanels allows the user to move the calendar panel to the right side of the control at runtime, if the user clicks the calendar panel while pressing the SHIFT + CTRL keys, and drag the mouse to the ride side of the schedule view. The <u>OnResizeControl</u> property is changed once the user drags a panel to a new position. The AllowExchangePanels property on exDisallow prevents exchanging the panels at runtime.

Also, you can use the OnResizeControl property to specify one of the followings:

- auto hide the calendar panel. Ability to hide the calendar section while the cursor is not in it (OnResizeControlEnum.exResizePanelRight Or OnResizeControlEnum.exCalendarFit Or OnResizeControlEnum.exCalendarAutoHide).
- hide completely the calendar section (exHideSplitter)
- specify the **alignment** of the calendar, as on the left or right side of the schedule view (OnResizeControlEnum.exChangePanels Or OnResizeControlEnum.exCalendarFit)
- full or partially view of the calendar panel (exResizePanelRight)
- disabling the control's vertical split bar (so user can not resize the fixed panel) (OnResizeControlEnum.exResizePanelRight Or OnResizeControlEnum.exDisableSplitter Or OnResizeControlEnum.exCalendarFit)

property Schedule.AllowMoveEvent as AllowKeysEnum

Specifies the combination of keys that allows the user to move the event.

Туре	Description
<u>AllowKeysEnum</u>	An AllowKeysEnum expression that specifies keys that allows the user to move the events from the cursor.

By default, the AllowMoveEvent property is exLeftClick, which indicates that the user can press the left mouse button inside an event or selection, to start moving one or more events. The AllowMoveEvent property on exDisallow indicates that user can not move the events. The AllowMoveEvent property indicates the combination of the keys to let user moves the events. The Movable property of the Event indicates whether an event can be moved at runtime. The MinDate/MaxDate properties of the Event indicates the lower or upper margins where the event can be moved. The UpdateEvent event occurs once an event is resized or moved. The AllowMoveEventToOtherGroup property indicates whether an event can be moved from a group to another. The AllowRefineMoveKey property specifies whether the margins of the events being updated are aligned to minor/major rulers of the control's time scale.

The <u>UpdateEventsLabel</u> property indicates the HTML format to be shown on the label when the user moves the events. The <u>UpdateEventsLabelAlign</u> property aligns the label being shown when the user moves the event. The

Background(exScheduleUpdateEventsBackColor) and

<u>Background</u>(exScheduleUpdateEventsForeColor) specifies the visual appearance of the event being moved. The <u>AllowCreateEvent</u> property indicates the combination of the keys to let user creates new events. The <u>AllowResizeEvent</u> property indicates the combination of the keys to let user resizes the events.

property Schedule.AllowMoveEventToOtherGroup as Boolean

Specifies if the event can ve moved from a group to another when dragging.

Туре	Description
Boolean	A Boolean expression that specifies whether the user can move an event from a group to another.

By default, the AllowMoveEventToOtherGroup property is True. The AllowMoveEventToOtherGroup property has effect only, if the control displays the events by groups (columns). The <u>GroupID</u> property of the Event specifies the identifier of the Group that hosts the event. If the control displays groups the GroupID property of the newly created event is automatically updated with the group where the event has been created. When the user moves an event from a group to another, at runtime, the GroupID property may be changed, and the <u>UpdateEvent</u> event occurs. The AllowMoveEventToOtherGroup property on False, prevents moving events from a group to another, at runtime.

The control displays groups if:

- <u>ShowGroupingEvents</u> property is True
- The <u>Groups</u> collection has elements. By default, the Groups collection contains no Group objects. Use the <u>Add</u> method of the Groups collection to add new groups to the control.
- The Groups collection contains visible elements (Visible on True)

property Schedule.AllowMoveGroup as AllowKeysEnum

Specifies the combination of keys that allows the user to move the group.

Туре	Description
<u>AllowKeysEnum</u>	An AllowKeysEnum expression that specifies the keys to change the Group's position at runtime.

By default, The AllowMoveGroup property is exLeftClick, which means the user can change the Group's position by dragging the Group's Header to a new position. The <u>Position</u> property can be used to programmatically change the Group's position by code. You can enumerate the Group as being displayed using the <u>ItemByPos</u> property of Groups collection. The <u>Visible</u> property of the Group specifies whether the Group is visible in the schedule view, and un-checked, in the drop down grouping panel. For instance, the AllowMoveGroup on exDisallow indicates that the user can not change the position of the group at runtime. The <u>AllowResizeGroup</u> property specifies whether the user can resize a group at runtime. The <u>LayoutStartChanging</u>(exScheduleMoveGroup) event occurs once the user starts moving a group. The <u>LayoutEndChanging</u>(exScheduleMoveGroup) event occurs the user moves a group.

property Schedule.AllowMoveMarkTime as AllowKeysEnum

Specifies the combination of keys that allows the user to move a mark time.

Туре	Description
<u>AllowKeysEnum</u>	An AllowKeysEnum expression that specifies the combination of keys the user can use so it can change the position/time of Movable timers.

By default, the AllowMoveMarkTime property is exLeftClick, which indicates that the user can move a timer by dragging it to a new position when clicking the left mouse button on a timer object. The AllowMoveMarkTime property indicates the keys to allow user to move timers (with the Movable property on True). The ShowMarkTime property indicates whether the schedule view displays timers. The MarkTimeFromPoint property indicates the timer from the cursor. The MarkTimes property gets a collection of MarkTime objects, also called timers. The MarkTime object indicates a line in the schedule view, at a specified time. The Add method of MarkTimes collection adds a new timer to the schedule view. The MarkTimes collection is accessible through the MarkTimes property of the control.

The MarkTime object, also called **timer**, can be used to:

- show a line of different styles on the schedule view, at specified time
- show a HTML label at specified time
- highlights the events that intersect with the timer

property Schedule.AllowMoveSchedule as AllowKeysEnum

Specifies the combination of keys that allows the user to move the schedule view.

Туре	Description
<u>AllowKeysEnum</u>	An AllowKeysEnum expression that specifies the keys combination so the user can navigate the entire calendar without selecting dates in the calendar panel.

By default, the AllowMoveSchedule property is exLeftClick + exSHIFTKey, which indicates that the user can navigate the entire scheduler by clicking the left mouse button and pressing the SHIFT key. The AllowMoveSchedule property allows the user to move or navigate the schedule view to a new position, without selecting a new date in the calendar panel. This options gives you the ability to go to neighbor view with just a click. The AllowResizeSchedule property allows you to magnify the schedule view to view more dates without selecting new dates in the calendar panel. The AllowExchangePanels property allows the user to move a panel from a side to another. The LayoutStartChanging(exScheduleMove) event occurs once the user starts navigating the schedule view to a new position. The LayoutEndChanging(exScheduleMove) event occurs once the user starts navigating the schedule view to a new position. The LayoutEndChanging(exScheduleMove) event occurs once the user starts navigating the schedule view to a new position. The LayoutEndChanging(exScheduleMove) event occurs once the user starts navigating the schedule view to a new position. The LayoutEndChanging(exScheduleMove) event occurs once the user starts navigating the schedule view to a new position. The LayoutEndChanging(exScheduleMove) event occurs once the user starts navigating the schedule view to a new position. The LayoutEndChanging(exScheduleMove) event occurs once the user starts navigating the schedule view to a new position. The LayoutEndChanging(exScheduleMove) event occurs once the user starts navigating the schedule view to a new position. The LayoutEndChanging(exScheduleMove) event occurs once the user starts navigating the schedule view to a new position. The LayoutEndChanging(exScheduleMove) event occurs once the user starts navigating the schedule view to a new position.

once the user moved the schedule view to a new position. The <u>FitSelToView</u> method restores the view to fit the selected dates. The <u>ClipToSel</u> property indicates whether the control clips the schedule panel to view the selected dates only.

The following screen shot shows the schedule view at initial position:



The following screen shot shows the schedule view at a new position:



property Schedule.AllowMoveTimeScale as AllowKeysEnum

Specifies the combination of keys to move the control's time scale.

Туре	Description
<u>AllowKeysEnum</u>	An AllowKeysEnum expression that indicates the keys the user can move at runtime the time scale from a side to another.

By default, the AllowMoveTimeScale property is exLeftClick + exSHIFTKey + exCTRLKey, which indicates that the user can move the position of the time scale, by dragging the mouse, if the user left clicking the mouse on the time scale and the SHIFT + CTRL keys are pressed. The AllowMoveTimeScale property indicates the keys the user can move at runtime the time scale from a side to another. The AlignLeft and Position properties may be changed if the user moves the time scale position to a new side, while the AllowMoveTimeScale property is not exDisallow (0). The AlignLeft property can be used to programmatically change the time scale alignment. The Position property indicates the position of the time scale as they are displayed.

The AllowMoveTimeScale property on exDisallow indicates that the user can not change the time scale position at runtime.

property Schedule.AllowMultiDaysEvent as Boolean

l

Specifies whether the user can create events that may start on a day and ends on other.

Туре	Description
Boolean	A Boolean expression that specifies whether the user can create multiple-days event.

By default, the AllowMultiDaysEvent property is True. The AllowMultiDaysEvent property indicates whether the user can create events that may start on a day and ends on other. The <u>AllowCreateAllDayEvent</u> property specifies whether the user can create all day events, at runtime. The <u>AddEvent</u> event is fired once the user creates a new event. The <u>UpdateEvent</u> event is fired once the margins of the event is being changed. The <u>Start</u> and <u>End</u> properties of the Event indicates the margins of the event. The AllowMultiDaysEvent property on False, prevents creating events that starts in a day and ends on other.

The <u>CreateEventLabel</u> property indicates the HTML format to be shown on the label when the user creates a new event. The <u>CreateEventLabelAlign</u> property aligns the label being shown when the user creates a new event.

property Schedule.AllowRefineMoveKey as AllowKeysEnum

Specifies the combination of keys to refine the start and end of events when creating, moving or resizing the events.

Туре	Description
<u>AllowKeysEnum</u>	An AllowKeysEnum expression that specifies the keys that allows the user to refine the creating, moving or resizing the events.

By default, the AllowRefineMoveKey property is exDisallow, which means that the property has no effect. The AllowRefineMoveKey property specifies whether the margins of the events being updated are aligned to minor/major rulers of the control's time scale. In other words, if the AllowRefineMoveKey property is exDisallow (by default), the margins of the events are always aligned to the minor/major rulers of the time scale when it is updated. If the AllowRefineMoveKey property is NOT exDisallow, the user can move, resize or create events at arbitrary times, not necessary aligned to the minor/major rulers of the time scale. The AllowCreateEvent property specifies the keys combination that allows the user to create events in the schedule view. The AllowMoveEvent property indicates the combination of the keys to let user moves the events. The AllowResizeEvent property indicates the combination of the keys to let user resizes the events.

For instance, you can use the AllowRefineMoveKey property on exLeftClick, so the margins of the events while updating are always arbitrary. Instead if you are using a combination such as exLeftClick + exCTRLKey the arbitrary margins are allowed for the updating events while the CTRL key is down, and aligned to the control's time scale(s) when no CTRL key is pressed.

property Schedule.AllowResizeEvent as AllowKeysEnum

Specifies the combination of keys that allows the user to resize the event.

Туре	Description
<u>AllowKeysEnum</u>	An AllowKeysEnum expression that specifies keys that allows the user to resize the events from the cursor.

By default, the AllowResizeEvent property is exLeftClick, which indicates that the user can press the left mouse button on the event's lower or upper margin, to start resizing one or more events (selection). The AllowResizeEvent property on exDisallow indicates that user can not resize the events. The AllowResizeEvent property indicates the combination of the keys to let user resizes the events. The <u>Resizable</u> property of the Event indicates whether an event can be resized at runtime. The <u>MinDate/MaxDate</u> properties of the Event indicates the lower or upper margins where the event can be resized. The <u>UpdateEvent</u> event occurs once an event is resized or moved. *The <u>AllowRefineMoveKey</u> property specifies whether the margins of the events being updated are aligned to minor/major rulers of the control's time scale.*

The <u>UpdateEventsLabel</u> property indicates the HTML format to be shown on the label when the user resizes the events. The <u>UpdateEventsLabelAlign</u> property aligns the label being shown when the user resizes the event. The

Background(exScheduleUpdateEventsBackColor) and

<u>Background</u>(exScheduleUpdateEventsForeColor) specifies the visual appearance of the event being moved. The <u>AllowCreateEvent</u> property indicates the combination of the keys to let user creates new events. The <u>AllowMoveEvent</u> property indicates the combination of the keys to let user moves the events.

property Schedule.AllowResizeGroup as AllowKeysEnum

Specifies the combination of keys that allows the user to resize the group.

Туре	Description
<u>AllowKeysEnum</u>	An AllowKeysEnum expression that specifies the keys combination so the user can resize a group.

By default, the AllowResizeGroup property is True, which means that the user can resize a group by left clicking the area between two groups, and dragging to a new position. The AllowResizeGroup property on exDisallow, prevents resizing the groups at runtime. The AllowMoveGroup property specifies whether the user can move a group from one position to another. The LayoutStartChanging(exScheduleResizeGroup) event occurs once the user starts resizing a group. The LayoutEndChanging(exScheduleResizeGroup) event occurs the user resizes a group. The GroupHeaderFromPoint property indicates the group from the cursor, in the header part. You can use the Width property to change the width of the Group by code.

property Schedule.AllowResizeSchedule as AllowKeysEnum

Specifies the combination of keys that allows the user to resize the schedule view.

Туре	Description
<u>AllowKeysEnum</u>	An AllowKeysEnum expression that specifies the keys combination so the user can magnify or shrink (zooming) the scheduler view.

By default, the AllowResizeSchedule property is exMiddleClick, which indicates that the user magnify or shrink (zooming) the scheduler view if dragging the mouse while the MIDDLE mouse button is pressed. The AllowResizeSchedule property allows the user to magnify or shrink (zooming) the scheduler view with a click. The AllowMoveSchedule property allows the user to move or navigate the schedule view to a new position, without selecting a new date in the calendar panel. The AllowExchangePanels property allows the user to move a panel from a side to another. The LayoutStartChanging(exScheduleResize) event occurs once the user starts resizing the schedule view. The LayoutEndChanging(exScheduleResize) event occurs once the user resized the schedule view. The FitSeIToView method restores the view to fit the selected dates. The ClipToSel property indicates whether the control clips the schedule panel to view the selected dates only.

The following screen shot shows the schedule view at initial position:



The following screen shot shows the schedule view being shrink:



property Schedule.AllowResizeTimeScale as AllowKeysEnum

Specifies the combination of keys that allows the user to resize the time scale.

Туре	Description
<u>AllowKeysEnum</u>	An AllowKeysEnum expression that indicates the keys the user can resize at runtime the time scale.

The AllowResizeTimeScale property indicates the keys the user can resize at runtime the time scale. The <u>AllowResize</u> property may specify whether a time scale is resizable or not, while AllowResizeTimeScale property on exDisallow may specify that no time scale can be resized at runtime. The <u>MinWidth</u> property indicates the minimum width for the time scale, and the <u>MaxWidth</u> indicates the maximum size of the time scale. The <u>Width</u> property of the time scale indicates the width in pixels of the time scale. The <u>LayoutStartChanging(</u> exScheduleResizeTimeScale) and <u>LayoutEndChanging(</u> exScheduleResizeTimeScale) properties notifies your application once the user resizes a time scale. The <u>Width</u> property may be changed while the user resizes the time scale.

property Schedule.AllowSelectCreateEvent as SelectCreateEventEnum

Specifies whether the newly created event gets selected or highlighted

Туре	Description
<u>SelectCreateEventEnum</u>	A <u>SelectCreateEventEnum</u> expression that specifies whether the control selects or highlights the newly created event.

By default, the AllowSelectCreateEvent property is exSelectCreateEventNone, which means that nothing happen if the user creates a new event. The AllowSelectCreateEvent property specifies whether the newly created event gets selected or highlighted. Use the ShowAllDayHeader property to show the schedule's All-Day header so all All-Day evens are shown on this header. The AllowCreateAllDayEvent property has effect only when the schedule displays no time scale. The AllDayEvent property indicates an all-day event. You can use the AllowSelectEvent property to change the key to allow the user select new events or you can prevent selecting any event using exDisallow value. The SelectEventColor / SelectEventTextColor indicates the background / foreground colors to be applied on the event's body when the control just highlights the event being created.

property Schedule.AllowSelectEvent as AllowKeysEnum

Specifies the combination of keys that allows the user to select events in the schedule panel.

Туре

Description

	An AllowKeysEnum expression that specifies the keys to
AllowReysEllull	allow the user to select events in the schedule view.

By default, the AllowSelectEvent property is exLeftClick, which means that the event gets selected as soon as the user clicks it (using the left mouse button). You can use the AllowSelectEvent property to change the key to allow the user select new events or you can prevent selecting any event using exDisallow value. The

LayoutStartChanging(exScheduleSelectionChange) event occurs once the user is about to change the selection (of events), in the schedule view. The LayoutEndChanging(exScheduleSelectionChange) event occurs once the user changed the selection (of events), in the schedule view. The <u>Selectable</u> property of the event indicates whether the event can be selected at runtime. The <u>Selected</u> property of the Event indicates whether the current event is selected or unselected. The <u>SelectEventStyle</u> property indicates the way the selected events are shown. The <u>SelectEventColor</u> property specifies the visual appearance of the selected event. The <u>SelectEventTextColor</u> property specifies the foreground color of the selected event. The <u>AllowSelectEventRect</u> property indicates whether the user can select multiple events by dragging a rectangle. The <u>AllowToggleSelectKey</u> property indicates the key to be used so the user can toggle a selected event.

The <u>Selection</u> property gets or sets a safe array of selected events. *The /NET or /WPF* version provides the SelEvents property of List<Event> type to get or sets the new selection using a collection of Event objects.

The AllowSelectEvent property on exDisallow specifies that no selection is allowed in the schedule view, so no event can be selected. Use the <u>Selectable</u> property of the Event on False, to specify whether the event is selectable or not.

property Schedule.AllowSelectEventRect as AllowKeysEnum

Specifies the combination of keys that allows the user to select events in the schedule panel, by dragging a rectangle.

Туре	Description
<u>AllowKeysEnum</u>	An AllowKeysEnum expression that specifies the keys to allow the user to select events using a dragging rectangle, in the schedule view.

By default, the AllowSelectEventRect property is exLeftClick + exALTKey, which indicates that the user can select multiple events by dragging a rectangle if the user clicks and keeps the ALT key pressed. You can use the <u>AllowSelectEvent</u> property to change the key to allow the user select new events or you can prevent selecting any event using exDisallow value. The <u>LayoutStartChanging</u>(exScheduleSelectionChange) event occurs once the user is about to change the selection (of events), in the schedule view. The <u>LayoutEndChanging</u>(exScheduleSelectionChange) event occurs once the user changed the selection (of events), in the schedule view. The <u>Selected</u> property of the event indicates whether the event can be selected at runtime. The <u>SelectEventStyle</u> property indicates the way the selected events are shown. The <u>SelectEventColor</u> property specifies the foreground color of the selected event. The <u>AllowToggleSelectKey</u> property indicates the key to be used so the user can toggle a selected event.

You can do the same type of the selection in the calendar panel, by using the <u>AllowSelectDateRect</u> property.

The following screen shot shows the rectangular selection, in the schedule panel:



property Schedule.AllowToggleSchedule as AllowKeysEnum

Toggles the schedule view when user double clicks it.

Туре	Description
<u>AllowKeysEnum</u>	An AllowKeysEnum expression that defines the keys to to toggle the schedule view.

By default, the AllowToggleSchedule property is exLeftClick + exDblClick, which means a double click will toggle the schedule view. The <u>AllowEditEvent</u> property uses the same keys combination as AllowToggleSchedule property, so if you double click an event, the inline editing is performed, else if clicking outside of the event the toggling the schedule view is performed. By toggling the schedule view we mean that for instance, if you select the entire month to be viewed, a double click on a date will bring the date to be shown on the schedule view, and the next double click will restore the month view. You can specify an event being not editable, using the <u>Editable</u> property on exNoEditable, or you can set the <u>AllowEditEvent</u> property on exDisallow, to prevent editing any of the schedule's events.

property Schedule.AllowToggleSelectKey as AllowKeysEnum

Specifies the combination of keys to select multiple not-contiguously events.

Туре	Description
<u>AllowKeysEnum</u>	An AllowKeysEnum expression that specifies the keys to allow the user to select/unselect an event, in the schedule view

By default, the AllowToggleSelectKey property is exCTRLKey, which indicates that the user can unselect/select an event by pressing the left mouse button and keeping the CTRL key down. The AllowToggleSelectKey property indicates the key to be used so the user can toggle a selected event. You can use the <u>AllowSelectEvent</u> property to change the key to allow the user select new events or you can prevent selecting any event using exDisallow value. The <u>AllowSelectEventRect</u> property indicates whether the user can select multiple events by dragging a rectangle.

The LayoutEndChanging(exScheduleSelectionChange) event occurs once the user changed the selection (of events), in the schedule view. The <u>Selectable</u> property of the event indicates whether the event can be selected at runtime. The <u>Selected</u> property of the Event indicates whether the current event is selected or unselected. The <u>SelectEventStyle</u> property indicates the way the selected events are shown. The <u>SelectEventColor</u> property specifies the visual appearance of the selected event. The <u>SelectEventTextColor</u> property specifies the foreground color of the selected event.

For instance, you can set the AllowToggleSelectKey property on exDisallow which indicates that no toggle selection is allowed. If the AllowToggleSelectKey property on exLeftClick, the first click selects the event, the next click unselect the event, and so on.

property Schedule.AllowUndoRedo as Boolean

Enables or disables the Undo/Redo feature.

Туре	Description
Boolean	A boolean expression that specifies whether the control supports Undo/Redofeature

By default, the AllowUndoRedo property is false, which indicates that the Undo/Redo feature us disabled. The Undo and Redo features let you remove or repeat single or multiple actions, but all actions must be undone or redone in the order you did or undid them you cant skip actions. For example, if you added three calendar-events and then decide you want to undo the first change you made, you must undo all three changes. To undo an action you need to press Ctrl+Z, while for to redo something you've undone, press Ctrl+Y. The <u>CanUndo</u> property retrieves a value that indicates whether the control may perform the last Undo operation. The <u>CanRedo</u> property retrieves a value that specifies whether the control can execute the next operation in the control's Redo queue. Call the <u>Undo</u> method to Undo the last control operation. The <u>Redo</u> redoes the next action in the control's redo queue. The <u>UndoRedoQueueLength</u> property gets or sets the maximum number of Undo/Redo actions that may be stored to the control's queue, or in other words how many operations the control's Undo/Redo manager may store.

The records of the Undo/Redo queue may contain actions in the following format:

- "AddEvent; EVENTID", indicates that a new calendar-event has been created
- "RemoveEvent; EVENTID", indicates that an calendar-event has been removed
- "MoveEvent; EVENTID", indicates that an calendar-event has been moved or resized
- "UpdateEvent; EVENTID", indicates that one or more properties of the calendar-event has been updated, using the <u>StartUpdateEvent</u> / <u>EndUpdateEvent</u> methods

Also, the Undo/Redo queue may include:

- "StartBlock", specifies that a block of operations begins (initiated by <u>StartBlockUndoRedo</u> method)
- "EndBlock", specifies that a block of operations ends (initiated by <u>EndBlockUndoRedo</u> method)

The LayoutStartChanging(exUndo/exRedo) / LayoutEndChanging(exUndo/exRedo) event notifies your application whenever an Undo/Redo operation is performed. The UndoListAction property lists the Undo actions that can be performed in the control. The RedoListAction property lists the Redo actions that can be performed in the control. Use the UndoRemoveAction method to remove the last actions from the undo queue. The RedoRemoveAction method removes the first action to be performed if the Redo method is invoked.

property Schedule.AllowUpdateAllDayFlag as Boolean

Indicates if the All-Day flag for the events being moved using drop and drop are updated once the user drops the selection.

Туре	Description
Boolean	A Boolean expression that specifies whether the <u>AllDayEvent</u> property is changed when the user drags and drops an event from All-Day header to Time-Scale section of the schedule view, and reverse.

By default, the AllowUpdateAllDayFlag property is True, which means that the <u>AllDayEvent</u> property of the event being dragged is updated once the user drops the event to All-Day header or Time-Scale part of the schedule view. The AllowUpdateAllDayFlag property has effect only if the <u>ShowAllDayHeader</u> property is True. The <u>AllowAllDayEventScroll</u> property gets or sets a value that specifies whether the all-day event header supports scrolling.

The following screen shot shows the All-Day events on the All-Day header (ShowAllDayHeader property is True):

	_	Sunday, May20, 12	Monday, May21, 12	Tuesday, May22, 12	Wednesday, May23, 12	Thursday, May 24, '12	Friday, May25, 12	Saturday, May26, 12
. 4		6/8/2012 - 5/31/2012						,
(C)		5/18/2012 - 5/23/2012				6/24/2012 - 5/25/2012 All-Day E	vent	M-Z
Õ		5/20/2012 - 5/23/2012				5/24/2012 - 5/25/2012 All-Dwy E	vent K-M	
<u> </u>				5/22/2012 - 5/23/2012			5/25/2012	
< 1					5/23/2012 - 5/24/2012 All-Day	Event A-K	5/25/2012 - 5/28/2012	
	08:00 AM				8:00 AM - 10:00 AM			
4	:20	In an and an an an						
	09:00 AM	8:40 AM - 12:40 PM	9.00 AM - 11:40 AM	9:00 AM - 1:20 PM	5. S.			
	:20							9.10 AM - 2:30 PM
	10-00 AM	-				10-00 AM - 12-00 PM		-
Ð	:20					TO AVE THE TRAVE THE	10:10 AM - 12:00 PM	
	:40		22					
8	11:00 AM				11:20 AM - 2:10 PM			
1	;40					dentist		
Le la	12:00 PM		12-20 PM - 2-30 PM				12:00 PM - 2:00 PM	
1	:40		12.20 18 12.00 18					
	01:00 PM	*****************************		1		1:00 PM - 3:10 PM		
	:20							
	02:00 PM							
	:20		•					
	03:00 PM					-		

property Schedule.AllowUpdateDisableZone as Boolean

Indicates whether the user can updates the events in the disabled part of the schedule.

Туре	Description
Boolean	A Boolean expression that specifies whether the

By default, the the AllowUpdateDisableZone property is False, so the user can not update or create new events in a disabled zone. The <u>DisableZoneFormat</u> property of the Calendar returns or sets an expression that determines the dates being disabled in the calendar/schedule panel. The AllowUpdateDisableZone property on True, lets user to update the disabled zones. The <u>AllowCreateEvent</u> property indicates the combination of the keys to let user creates new events. The <u>AllowMoveEvent</u> property indicates the combination of the keys to let user moves the events. The <u>AllowResizeEvent</u> property indicates the combination of the keys to let user resizes the events.

The following screen shot shows all dates as disabled (the entire month is selected):



DisableZoneFormat = "1"

A disable zone, shows as grayed as in the following screen shot (only dates: 23, 24, and 25 are enabled, and the rest are disabled, the entire month is selected):

		May 1	2] 3]	4	5	6]^
			•					
May 2012	7	8	9	10	11	12	13	ĩ
Mo Tu We Th Fr Sa Su 18 1 2 3 4 5 6								-
19 7 8 9 10 11 12 13	14	15	16	17	18	19	20	12
20 14 15 16 17 18 19 20 21 21 22 23 24 25 26 27					_			
22 28 29 30 31	21	22	23	24	25	26	27	Ĩ
23								
	28	29	30	May 31				
Today		-						-

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

Retrieves the identifier of the anchor from point.

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

the AnchorFromPoint property to determine the identifier of the anchor from the point. Use the <a id;options> anchor elements to add hyperlinks to cell's caption. The control fires the <u>AnchorClick</u> event when the user clicks an anchor element. Use the <u>ShowToolTip</u> method to show the specified tooltip at given or cursor coordinates. The <u>MouseMove</u> event is generated continually as the mouse pointer moves across the control. For instance, if the user clicks the anchor <a1>anchor, the control fires the AnchorClick event, where the AnchorID parameter is 1, and the Options parameter is empty. Also, if the user clicks the anchor <a 1;yourextradata>anchor, the AnchorID parameter of the AnchorClick event is 1, and the Options parameter.

The LongLabel, ExtraLabel properties of the Event supports <a> elements.

property Schedule.Appearance as AppearanceEnum

Retrieves or sets the control's appearance.

Туре	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 <u>Appearance</u> collection, being displayed as control's borders. For instance, if the Appearance = 0x1000000, indicates that the first skin object in the Appearance collection defines the control's border. The Client object in the skin, defines the client area of the control. The list/hierarchy/chart, scrollbars are always shown in the control's client area. The skin may contain transparent objects, and so you can define round corners. The <u>hot.ebn</u> file contains such of objects. Use the <u>eXButton</u> 's Skin builder to view or change this file

Use the Appearance property to specify the control's border. Use the <u>Add</u> method to add new skins to the control. Use the <u>BackColor</u> property to specify the control's background color. Use the <u>Background(exToolTipAppearance)</u> property indicates the visual appearance of the borders of the tooltips.



The following samples shows the control's borders using the <u>hot.ebn</u> file:

VBA (MS Access, Excell...)

```
With Schedule1
.VisualAppearance.Add 1,"c:\exontrol\images\hot.ebn"
.Appearance = 16777216
End With
```

VB6

```
With Schedule1
.VisualAppearance.Add 1,"c:\exontrol\images\hot.ebn"
.Appearance = &H1000000
End With
```

VB.NET

With Exschedule1 .VisualAppearance.Add(1,"c:\exontrol\images\hot.ebn") .Appearance = &H1000000 End With

VB.NET for /COM

```
With AxSchedule1
.VisualAppearance.Add(1,"c:\exontrol\images\hot.ebn")
.Appearance = &H1000000
End With
```

C++

```
/*
  Copy and paste the following directives to your header file as
  it defines the namespace 'EXSCHEDULELib' for the library: 'ExSchedule 1.0 Control
Library'
  #import <ExSchedule.dll>
  using namespace EXSCHEDULELib;
 */
EXSCHEDULELib::ISchedulePtr spSchedule1 = GetDlgItem(IDC_SCHEDULE1)-
 >GetControlUnknown();
spSchedule1->GetVisualAppearance()->Add(1,"c:\\exontrol\\images\\hot.ebn");
```

```
spSchedule1->PutAppearance(EXSCHEDULELib::AppearanceEnum(0x100000));
```

C++ Builder

```
Schedule1->VisualAppearance->Add(1,TVariant("c:\\exontrol\\images\\hot.ebn"));
Schedule1->Appearance = Exschedulelib_tlb::AppearanceEnum(0x1000000);
```

C#

```
exschedule1.VisualAppearance.Add(1,"c:\\exontrol\\images\\hot.ebn");
exschedule1.Appearance = (exontrol.EXSCHEDULELib.AppearanceEnum)0x1000000;
```

JavaScript

```
<OBJECT classid="clsid:9B09E13D-7A88-4299-9DBE-383380435377"
id="Schedule1"> </OBJECT>
<SCRIPT LANGUAGE="JScript">
Schedule1.VisualAppearance.Add(1,"c:\\exontrol\\images\\hot.ebn");
Schedule1.Appearance = 16777216;
</SCRIPT>
```

C# for /COM

axSchedule1.**VisualAppearance**.Add(1,"c:\\exontrol\\images\\hot.ebn"); axSchedule1.**Appearance** = (EXSCHEDULELib.AppearanceEnum)0x1000000;

X++ (Dynamics Ax 2009)

```
public void init()
{
;
super();
```

```
exschedule1.VisualAppearance().Add(1,"c:\\exontrol\\images\\hot.ebn");
exschedule1.Appearance(16777216);
```

Delphi 8 (.NET only)

```
with AxSchedule1 do
begin
VisualAppearance.Add(1,'c:\exontrol\images\hot.ebn');
Appearance := EXSCHEDULELib.AppearanceEnum($1000000);
end
```

Delphi (standard)

```
with Schedule1 do
begin
VisualAppearance.Add(1,'c:\exontrol\images\hot.ebn');
Appearance := EXSCHEDULELib_TLB.AppearanceEnum($100000);
end
```

VFP

with thisform.Schedule1
.VisualAppearance.Add(1,"c:\exontrol\images\hot.ebn")
.Appearance = 16777216
endwith

dBASE Plus

```
local oSchedule
```

oSchedule = form.Activex1.nativeObject
oSchedule.VisualAppearance.Add(1,"c:\exontrol\images\hot.ebn")
oSchedule.Appearance = 16777216 /*0x1000000 | */

XBasic (Alpha Five)

Dim oSchedule as P
oSchedule = topparent:CONTROL_ACTIVEX1.activex
oSchedule.VisualAppearance.Add(1,"c:\exontrol\images\hot.ebn")
oSchedule.Appearance = 16777216 '16777216 +

Visual Objects

oDCOCX_Exontrol1:**VisualAppearance**:Add(1,"c:\exontrol\images\hot.ebn") oDCOCX_Exontrol1:**Appearance** := 0x1000000 |

PowerBuilder

OleObject oSchedule

oSchedule = ole_1.Object
oSchedule.VisualAppearance.Add(1,"c:\exontrol\images\hot.ebn")
oSchedule.Appearance = 16777216 /*16777216 /*0x100000*/ | */

property Schedule.ApplyGroupingColors as Boolean

Specifies whether the schedule view shows the events using the colors of owner groups.

Туре	Description
Boolean	A Boolean expression that specifies whether the events change the colors based on the owner groups.

By default, the ApplyGroupingColors property is True. Use the ApplyGroupingColors property to prevent showing the events with different colors specified by the groups' <u>EventForeColor</u> and <u>EventBackColor</u> properties. The ApplyGroupingColors property indicates whether the <u>EventForeColor</u> and <u>EventBackColor</u>, properties of the Group object are being applied to events. The <u>DisplayGroupingButton</u> property shows or hides the grouping button being displayed in the date's header. The <u>ShowGroupingEvents</u> property indicates whether the events are displayed on groups columns. The <u>SingleGroupingView</u> property specifies whether the drop down panel shows radio buttons, instead check boxes, so the user can see all groups or only one group at the time. The <u>GroupID</u> property specifies the identifier of the group where the event belongs. If the control displays groups the GroupID property of the newly created event is automatically updated with the group where the event has been created. The <u>AllowMoveEventToOtherGroup</u> property specifies whether the user can move an event from a group to another at runtime. Use the <u>Add</u> method of the Groups collection to add new groups to the control.

method Schedule.AttachTemplate (Template as Variant)

Attaches a script to the current object, including the events, from a string, file, a safe array of bytes.

Туре	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 Schedule1_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>"]"#"
<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 <u>Template</u> / <u>ExecuteTemplate</u> is that the AttachTemplate can add handlers to the control events.

property Schedule.BackColor as Color

Specifies the control's background color.

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

The BackColor property changes the control's background color as well as the calendar's background color, if the Background(exCalendarBackColor) is zero (by default). The Background(exCalendarBackColor) property changes the calendar's panel backcolor if not-zero. The Background(exCalendarForeColor) property changes the calendar's panel foreground color. The ForeColor property changes the control's foreground color. Use the Picture property of the control to show a picture on the control's background. The BodyEventBackColor property specifies the background color to show the body for all events. The EventBackColor property specifies the event's background color if it belongs to a group.

property Schedule.Background(Part as BackgroundPartEnum) as Color

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

Туре	Description
Part as <u>BackgroundPartEnum</u>	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 <u>Add</u> method to add new skins to the control. Use the <u>Remove</u> method to remove a specific skin from the control. Use the <u>Clear</u> method to remove all skins in the control. Use the <u>BeginUpdate</u> and <u>EndUpdate</u> methods to maintain performance while init the control. Use the <u>Refresh</u> method to refresh the control.

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

		Wednesday, June 20, 2	2012	
08:00 AM :30	8:15 AM - 10:30 AM]		
09:00 AM		9:00 AM - 12:30 PM		
:30				
10:00 AM		J		
:30				
11:00 AM	11:00 AM - 12:15 PM			
:30			(11.15 ML 0.15 ML	<u> </u>
12:00 PM			11:40 AM - 2:10 PM	
:30				
01:00 PM	C			
:30	1:15 PM - 2:30 PM			
02:00 PM				2
:30				-
03:00 PM	2:45 PM - 3:30 PM			
:30				

The following samples changes the background color for selected dates in the calendar panel, so always is the same no matter if the control loses the focus.

VBA (MS Access, Excell...)

```
With Schedule1

.Background(68) = .Background(19)

.Background(69) = .Background(20)

End With
```

VB6

```
With Schedule1

.Background(exCalendarSelBackColorUnFocus) =

.Background(exCalendarSelBackColor)

.Background(exCalendarSelForeColorUnFocus) =

.Background(exCalendarSelForeColor)

End With
```

VB.NET

With Exschedule1

.set_Background(exontrol.EXSCHEDULELib.BackgroundPartEnum.exCalendarSelBackCo

.set_Background(exontrol.EXSCHEDULELib.BackgroundPartEnum.exCalendarSelForeCo

End With

VB.NET for /COM

With AxSchedule1

.set_Background(EXSCHEDULELib.BackgroundPartEnum.exCalendarSelBackColorUnFc

.set_Background(EXSCHEDULELib.BackgroundPartEnum.exCalendarSelForeColorUnFo

End With

C++

```
Copy and paste the following directives to your header file as
it defines the namespace 'EXSCHEDULELib' for the library: 'ExSchedule 1.0 Control
Library'
```

```
#import <ExSchedule.dll>
using namespace EXSCHEDULELib;
```

```
*/
```

```
EXSCHEDULELib::ISchedulePtr spSchedule1 = GetDlgItem(IDC_SCHEDULE1)-
```

```
>GetControlUnknown();
```

```
spSchedule1-
```

```
>PutBackground(EXSCHEDULELib::exCalendarSelBackColorUnFocus,spSchedule1-
>GetBackground(EXSCHEDULELib::exCalendarSelBackColor));
```

spSchedule1-

> PutBackground (EXSCHEDULELib::exCalendarSelForeColorUnFocus, spSchedule1-

```
> GetBackground(EXSCHEDULELib::exCalendarSelForeColor));
```

C++ Builder

```
Schedule1-
```

```
> Background[Exschedulelib_tlb::BackgroundPartEnum::exCalendarSelBackColorUnFoc
```

```
= Schedule1-
```

>Background[Exschedulelib_tlb::BackgroundPartEnum::exCalendarSelBackColor]; Schedule1-

```
>Background[Exschedulelib_tlb::BackgroundPartEnum::exCalendarSelForeColorUnFoc
= Schedule1-
```

```
> Background[Exschedulelib_tlb::BackgroundPartEnum::exCalendarSelForeColor];
```

C#

exschedule1.set_Background(exontrol.EXSCHEDULELib.BackgroundPartEnum.exCalence

exschedule1.set_Background(exontrol.EXSCHEDULELib.BackgroundPartEnum.exCalence

JavaScript

```
<OBJECT classid="clsid:9B09E13D-7A88-4299-9DBE-383380435377"
id="Schedule1"> </OBJECT>
<SCRIPT LANGUAGE="JScript">
Schedule1.Background(68) = Schedule1.Background(19);
Schedule1.Background(69) = Schedule1.Background(20);
</SCRIPT>
```

C# for /COM

axSchedule1.**set_Background**(EXSCHEDULELib.BackgroundPartEnum.exCalendarSelBackground) axSchedule1.**set_Background**(EXSCHEDULELib.BackgroundPartEnum.exCalendarSelFo

X++ (Dynamics Ax 2009)

```
public void init()
{
;
super();
exschedule1.Background(68/*exCalendarSelBackColorUnFocus*/,exschedule1.Backgrc
exschedule1.Background(69/*exCalendarSelForeColorUnFocus*/,exschedule1.Backgrc
}
```

Delphi 8 (.NET only)

```
with AxSchedule1 do begin
```

 $set_Background ({\sf EXSCHEDULELib}. {\sf BackgroundPartEnum}. exCalendarSelBackColorUnFormatter and {\sf BackgroundPartEnum}. exCalendarSelBackGroundPartEnum And {\sf BackgroundPartEnum}. exC$

set_Background(EXSCHEDULELib.BackgroundPartEnum.exCalendarSelForeColorUnFo

end

Delphi (standard)

```
with Schedule1 do
begin
Background[EXSCHEDULELib_TLB.exCalendarSelBackColorUnFocus] :=
Background[EXSCHEDULELib_TLB.exCalendarSelBackColor];
Background[EXSCHEDULELib_TLB.exCalendarSelForeColorUnFocus] :=
Background[EXSCHEDULELib_TLB.exCalendarSelForeColor];
end
```

VFP

```
with thisform.Schedule1
  .Object.Background(68) = .Background(19)
  .Object.Background(69) = .Background(20)
endwith
```

dBASE Plus

```
local oSchedule
```

oSchedule = form.Activex1.nativeObject oSchedule.Template = [Background(68) = Background(19)] // oSchedule.Background(68) = oSchedule.Background(19) oSchedule.Template = [Background(69) = Background(20)] // oSchedule.Background(69) = oSchedule.Background(20)

XBasic (Alpha Five)

Dim oSchedule as P

oSchedule = topparent:CONTROL_ACTIVEX1.activex

oSchedule.Template = "Background(68) = Background(19)" '
oSchedule.Background(68) = oSchedule.Background(19)
oSchedule.Template = "Background(69) = Background(20)" '
oSchedule.Background(69) = oSchedule.Background(20)

Visual Objects

oDCOCX_Exontrol1:[Background,exCalendarSelBackColorUnFocus] := oDCOCX_Exontrol1:[Background,exCalendarSelBackColor] oDCOCX_Exontrol1:[Background,exCalendarSelForeColorUnFocus] := oDCOCX_Exontrol1:[Background,exCalendarSelForeColor]

PowerBuilder

OleObject oSchedule

oSchedule = ole_1.Object
oSchedule.Background(68,oSchedule.Background(19))
oSchedule.Background(69,oSchedule.Background(20))

method Schedule.BeginUpdate ()

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

Туре

Description

This method prevents the control from painting until the EndUpdate method is called. The BeginUpdate and <u>EndUpdate</u> methods increases the speed of loading your events, by preventing painting the control when it suffers any change. Once that BeginUpdate method was called, you have to make sure that EndUpdate method will be called too. You can use the <u>Refresh</u> method to refresh the control's content.

property Schedule.BodyEventBackColor as Color

Specifies the default visual appearance of the events.

Туре	Description
Color	A Color expression that specifies the background color to show the event's body. The last 7 bits in the high significant byte of the color to indicates the identifier of the skin being used. Use the <u>Add</u> 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 BodyEventBackColor property specifies the background color to show the body for all events. The BodyBackColor property specifies the background color of the event's body. The EventBackColor property specifies the event's background color if it belongs to a group. The BodyForeColor property specifies the foreground color to show the labels on the event. The BodyPattern property gives access to the pattern to be shown on the event's body. The StatusColor property indicates the color show the event's status. The ShowEvents property specifies what events the control should show. The EventsTransparent property indicates the transparency to show the events on the schedule view.

property Schedule.BodyEventForeColor as Color

Specifies the default foreground color of the events.

Туре	Description
Color	A Color expression that specifies the foreground color to show the event's body.

The BodyEventForeColor property specifies the foreground color to show the body for all events. The <u>BodyForeColor</u> property specifies the foreground color of the event's body. The <u>EventForeColor</u> property specifies the event's foreground color if it belongs to a group. The <u>BodyBackColor</u> property specifies the background color of the event's body. The <u>EventBackColor</u> property specifies the event's background color if it belongs to a group. The <u>StatusColor</u> property indicates the color show the event's status. The <u>ShowEvents</u> property specifies what events the control should show. The <u>EventsTransparent</u> property indicates the transparency to show the events on the schedule view.

property Schedule.BorderDateStyle as LinesStyleEnum

Specifies the style to display the border for the dates.

Туре	Description
<u>LinesStyleEnum</u>	A LinesStyleEnum expression that specifies whether the lines are shown between dates.

By default, the BorderDateStyle property specifies the lines to be shown between dates, in the schedule view. The <u>Background</u> property indicates the color to show the border lines. For instance, the Background(exScheduleBorderSelColor),

Background(exScheduleBorderSelColorUnFocus) indicates the color to show the border around the selection dates in the schedule view.

The control supports the following border properties:

- BorderDateStyle property specifies the lines to be shown between dates
- <u>BorderGroupStyle</u> property indicates the lines between groups. This property has effect only, if the schedule view displaying groups.
- <u>BorderMonthStyle</u> property specifies the style of lines to show the margins of the month. The effect of this property can be seen when entire month is shown in the schedule view.
- <u>BorderSelStyle</u> property specifies the type of lines to be shown around the selected dates. Use the BorderSelStyle property on exNoLines to show no lines for any selected date in the schedule view.
- <u>BorderTimeScaleStyle</u> property specifies the lines to delimit the control's time scales.

property Schedule.BorderGroupStyle as LinesStyleEnum

Specifies the style to display the border between groups within the date.

Туре	Description
LinesStyleEnum	A LinesStyleEnum expression that specifies whether the lines are shown between groups.

The BorderGroupStyle property indicates the lines between groups, in the schedule view. This property has effect only, if the schedule view displaying groups. The <u>Background</u> property indicates the color to show the border lines. For instance, the Background(exScheduleBorderSelColor), Background(exScheduleBorderSelColorUnFocus) indicates the color to show the border around the selection dates in the schedule view.

The control supports the following border properties:

- BorderDateStyle property specifies the lines to be shown between dates
- BorderGroupStyle property indicates the lines between groups. This property has effect only, if the schedule view displaying groups.
- <u>BorderMonthStyle</u> property specifies the style of lines to show the margins of the month. The effect of this property can be seen when entire month is shown in the schedule view.
- <u>BorderSelStyle</u> property specifies the type of lines to be shown around the selected dates. Use the BorderSelStyle property on exNoLines to show no lines for any selected date in the schedule view.
- <u>BorderTimeScaleStyle</u> property specifies the lines to delimit the control's time scales.

property Schedule.BorderHeight as Long

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

Туре	Description
Long	A long expression that specifies the height of the border being applied to the top and bottom side of the control.

By default, the BorderHeight property is 0. The BorderHeight property specifies the height of the border in pixels, being applied to the top and bottom side of the control. The <u>BorderWidth</u> property specifies the width of the border on the left and right side of the control. The borders delimit the margin of the control and the client area, where the calendar and the schedule panel is displayed.

property Schedule.BorderMonthStyle as LinesStyleEnum

Specifies the style to display the border for the months.

Туре	Description
<u>LinesStyleEnum</u>	A LinesStyleEnum expression that specifies the style of lines to show the margins of the month.

By default, the BorderMonthStyle property specifies the style of lines to show the margins of the month, in the schedule view. The effect of this property can be seen when entire month is shown in the schedule view. The <u>Background</u> property indicates the color to show the border lines. For instance, the Background(exScheduleBorderSelColor), Background(exScheduleBorderSelColorUnFocus) indicates the color to show the border around the selection dates in the schedule view.

The control supports the following border properties:

- BorderDateStyle property specifies the lines to be shown between dates
- <u>BorderGroupStyle</u> property indicates the lines between groups. This property has effect only, if the schedule view displaying groups.
- BorderMonthStyle property specifies the style of lines to show the margins of the month.
- <u>BorderSelStyle</u> property specifies the type of lines to be shown around the selected dates. Use the BorderSelStyle property on exNoLines to show no lines for any selected date in the schedule view.
- <u>BorderTimeScaleStyle</u> property specifies the lines to delimit the control's time scales.

property Schedule.BorderSelStyle as LinesStyleEnum

Specifies the style to display the border for selected dates.

Туре	Description
LinesStyleEnum	A LinesStyleEnum expression that specifies the type of lines to be shown around the selected dates.

By default, the BorderSelStyle property specifies the type of lines to be shown around the selected dates, in the schedule view. *Use the BorderSelStyle property on exNoLines to show no lines for any selected date in the schedule view.* The <u>Background</u> property indicates the color to show the border lines. For instance, the Background(exScheduleBorderSelColor), Background(exScheduleBorderSelColorUnFocus)

indicates the color to show the border around the selection dates in the schedule view.

The control supports the following border properties:

- BorderDateStyle property specifies the lines to be shown between dates
- <u>BorderGroupStyle</u> property indicates the lines between groups. This property has effect only, if the schedule view displaying groups.
- <u>BorderMonthStyle</u> property specifies the style of lines to show the margins of the month. The effect of this property can be seen when entire month is shown in the schedule view.
- BorderSelStyle property specifies the type of lines to be shown around the selected dates.
- <u>BorderTimeScaleStyle</u> property specifies the lines to delimit the control's time scales.

property Schedule.BorderTimeScaleStyle as LinesStyleEnum

Specifies the style to display the border for time scales.

Туре	Description
<u>LinesStyleEnum</u>	A LinesStyleEnum expression that specifies the lines to delimit the control's time scales.

By default, the <u>BorderTimeScaleStyle</u> property specifies the lines to delimit the control's time scales, in the schedule view. The <u>Background</u> property indicates the color to show the border lines. For instance, the Background(exScheduleBorderSelColor), Background(exScheduleBorderSelColorUnFocus) indicates the color to show the border around the selection dates in the schedule view.

The control supports the following border properties:

- BorderDateStyle property specifies the lines to be shown between dates
- <u>BorderGroupStyle</u> property indicates the lines between groups. This property has effect only, if the schedule view displaying groups.
- <u>BorderMonthStyle</u> property specifies the style of lines to show the margins of the month. The effect of this property can be seen when entire month is shown in the schedule view.
- <u>BorderSelStyle</u> property specifies the type of lines to be shown around the selected dates. Use the BorderSelStyle property on exNoLines to show no lines for any selected date in the schedule view.
- BorderTimeScaleStyle property specifies the lines to delimit the control's time scales.

property Schedule.BorderWidth as Long

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

Туре	Description
Long	A long expression that specifies the width of the border being applied to the left and right side of the control.

By default, the BorderWidth property is 0. The BorderWidth property specifies the width of the border in pixels, being applied to the left and right side of the control. The <u>BorderHeight</u> property specifies the height of the border on the top and bottom side of the control. The borders delimit the margin of the control and the client area, where the calendar and the schedule panel is displayed.

property Schedule.Calendar as Calendar

Gets the schedule's calendar object.

Туре	Description
<u>Calendar</u>	A Calendar object that holds information about the calendar panel of the control.

The Calendar property gives access to the calendar panel of the control. For instance, the <u>FirstWeekDay</u> property specifies the first day of the week. The <u>SingleSel</u> property indicates whether the user can select one or multiple dates. As an alternative, you can use the <u>SelCount/SelDate</u> property to retrieve the collection of selected dates in the calendar panel. Once the user starts selecting a new date in the calendar panel, the control fires the <u>LayoutStartChanging</u>(exCalendarSelectionChange). Once a new date is selected, the <u>LayoutEndChanging</u>(exCalendarSelectionChange) event occurs.

Use the <u>OnResizeControl</u> property to specify one of the followings:

- auto hide the calendar panel. Ability to hide the calendar section while the cursor is not in it (OnResizeControlEnum.exResizePanelRight Or OnResizeControlEnum.exCalendarFit Or OnResizeControlEnum.exCalendarAutoHide).
- hide completely the calendar section (exHideSplitter)
- specify the alignment of the calendar, as on the left or right side of the schedule view (OnResizeControlEnum.exChangePanels Or OnResizeControlEnum.exCalendarFit)
- full or partially view of the calendar panel (exResizePanelRight)

property Schedule.CanRedo as Boolean

Retrieves a value that indicates whether the control can perform a Redo operation.

Туре	Description
Boolean	A boolean expression that specifies whether the control can perform a Redo operation

The CanRedo method indicates whether the control can perform a Redo operation. The <u>AllowUndoRedo</u> property enables or disables the Undo/Redo feature. The <u>Redo</u> redoes the next action in the control's redo queue. The <u>Undo</u> method undoes the last control operation. The <u>UndoRedoQueueLength</u> property gets or sets the maximum number of Undo/Redo actions that may be stored to the control's queue, or in other words how many operations the control's Undo/Redo manager may store.

The records of the Undo/Redo queue may contain actions in the following format:

- "AddEvent; EVENTID", indicates that a new calendar-event has been created
- "RemoveEvent; EVENTID", indicates that an calendar-event has been removed
- "MoveEvent; EVENTID", indicates that an calendar-event has been moved or resized
- "UpdateEvent; EVENTID", indicates that one or more properties of the calendar-event has been updated, using the <u>StartUpdateEvent</u> / <u>EndUpdateEvent</u> methods

Also, the Undo/Redo queue may include:

- "StartBlock", specifies that a block of operations begins
- "EndBlock", specifies that a block of operations ends

The LayoutStartChanging(exUndo/exRedo) / LayoutEndChanging(exUndo/exRedo) event notifies your application whenever an Undo/Redo operation is performed. The UndoListAction property lists the Undo actions that can be performed in the control. The RedoListAction property lists the Redo actions that can be performed in the control. Use the UndoRemoveAction method to remove the last actions from the undo queue. The RedoRemoveAction method removes the first action to be performed if the Redo method is invoked.

property Schedule.CanUndo as Boolean

Retrieves a value that indicates whether the control can perform an Undo operation.

Туре	Description
Boolean	A boolean expression that specifies whether the control can perform an Undo operation

The CanUndo method indicates whether the control can perform an Undo operation. The <u>AllowUndoRedo</u> property enables or disables the Undo/Redo feature. The <u>Undo</u> method undoes the last control operation. The <u>Redo</u> redoes the next action in the control's redo queue. The <u>UndoRedoQueueLength</u> property gets or sets the maximum number of Undo/Redo actions that may be stored to the control's queue, or in other words how many operations the control's Undo/Redo manager may store.

The records of the Undo/Redo queue may contain actions in the following format:

- "AddEvent; EVENTID", indicates that a new calendar-event has been created
- "RemoveEvent; EVENTID", indicates that an calendar-event has been removed
- "MoveEvent; EVENTID", indicates that an calendar-event has been moved or resized
- "UpdateEvent; EVENTID", indicates that one or more properties of the calendar-event has been updated, using the <u>StartUpdateEvent</u> / <u>EndUpdateEvent</u> methods

Also, the Undo/Redo queue may include:

- "StartBlock", specifies that a block of operations begins
- "EndBlock", specifies that a block of operations ends

The LayoutStartChanging(exUndo/exRedo) / LayoutEndChanging(exUndo/exRedo) event notifies your application whenever an Undo/Redo operation is performed. The UndoListAction property lists the Undo actions that can be performed in the control. The RedoListAction property lists the Redo actions that can be performed in the control. Use the UndoRemoveAction method to remove the last actions from the undo queue. The RedoRemoveAction method removes the first action to be performed if the Redo method is invoked.

method Schedule.ClearAll ()

Clears all control's collections, including the events.

Туре

Description

The ClearAll method clears all control's collection as listed bellow. Usually, you can call this method prior to a <u>LoadXML</u> method for a fresh and empty control.

The ClearAll method clears the following collections:

- <u>TimeScales</u> collection. The TimeScales collection can be accessed through the component's <u>TimeScales</u> property. This is a particular case, when not the entire collection is emptied, as the first element in the collection, is not removed when the ClearAll method is called. In other words, the default TimeScale object being displayed, is not removed, just any additional TimeScale objects. You can use the <u>Clear</u> method of the TimeScales collection to remove all TimeScales objects. As the default, TimeScale object is not being removed or deleted, the TimeZone or any other property that has been changed, will not be changed to its default value.
- <u>Groups</u> collection. The Groups collection can be accessed through the component's <u>Groups</u> property.
- <u>NonworkingPatterns</u> collection. The NonworkingPatterns collection can be accessed through the component's <u>NonworkingPatterns</u> property.
- <u>NonworkingTimes</u> collection. The NonworkingTimes collection can be accessed through the component's <u>NonworkingTimes</u> property.
- <u>MarkZones</u> collection. The MarkZones collection can be accessed through the component's <u>MarkZones</u> property.
- <u>MarkTimes</u> collection. The MarkTimes collection can be accessed through the component's <u>MarkTimes</u> property.
- <u>ExPictures</u> collection. The ExPictures collection can be accessed through the component's <u>Pictures</u> property.
- Images/Icons list. The Images list contains the icons of the component, and can be accessed through the <u>Images</u> method.
- <u>Events</u> collection. The Events collection can be accessed through the component's <u>Events</u> property.

property Schedule.ClipToSel as Boolean

Specifies whether the schedule view displays the selection only.

Туре	Description
Boolean	A boolean expression that specifies whether the schedule view displays the selected dates only.

The ClipToSel property indicates whether the control clips the schedule panel to view the selected dates only. The <u>FitSelToView</u> method restores the view to fit the selected dates. Use the <u>Selection/SelectDate</u> property to change programmatically the dates being selected in the calendar, including the dates to be shown in the schedule view. You can use the <u>Selection/SelCount/SelDate</u> property to retrieve the selected dates. The <u>AllowResizeSchedule</u> property allows you to magnify the schedule view to view more dates without selecting new dates in the calendar panel. The <u>AllowMoveSchedule</u> property allows the user to move or navigate the schedule view to a new position, without selecting a new date in the calendar panel.

Once the user starts selecting a new date in the calendar panel, the control fires the <u>LayoutStartChanging</u>(exCalendarSelectionChange). Once a new date is selected, the <u>LayoutEndChanging</u>(exCalendarSelectionChange) event occurs.

method Schedule.Copy()

Copies the control's content to the clipboard, in the EMF format.

Туре

Description

Use the Copy method to copy the control's content to the clipboard, in Enhanced Metafile (EMF) format. The Enhanced Metafile format is a 32-bit format that can contain both vector information and bitmap information. This format is an improvement over the Windows Metafile Format and contains extended features, such as the following:

Built-in scaling information Built-in descriptions that are saved with the file Improvements in color palettes and device independence

The EMF format is an extensible format, which means that a programmer can modify the original specification to add functionality or to meet specific needs. You can paste this format to Microsoft Word, Excel, Front Page, Microsoft Image Composer and any application that know to handle EMF formats

The <u>CopyTo</u> method copies the control's content to a file. The <u>SaveXML</u> method saves the control's data to a file.

property Schedule.CopyTo (File as String) as Variant

Exports the control's view to an EMF file.

Туре	Description
	A String expression that indicates the name of the file to be saved. If present, the CopyTo property retrieves True, if the operation succeeded, else False it is failed. If the File parameter is missing or empty, the CopyTo property retrieves an one dimension safe array of bytes that contains the EMF content.
	If the File parameter is not empty, the extension (characters after last dot) determines the graphical/ format of the file to be saved as follows:
File as String	 *.bmp *.dib *.rle, saves the control's content in BMP format. *.jpg *.jpe *.jpeg *.jfif, saves the control's content in JPEG format. *.gif, , saves the control's content in GIF format. *.tif *.tiff, saves the control's content in TIFF format. *.png, saves the control's content in PNG format. *.pdf, saves the control's content to PDF format. The File argument may carry up to 4 parameters separated by the character in the following order: <i>filename.pdf</i> <i>paper size</i> <i>margins</i> <i>options</i>. In other words, you can specify the file name of the PDF document, the paper size, the margins and options to build the PDF document. By default, the paper size is 210 mm × 297 mm (A4 format) and the margins are 12.7 mm 12.7 mm 12.7 mm 12.7 mm. The units for the paper size and margins can be pt for PostScript Points, mm for Millimeters, cm for Centimeters, in for Inches and px for pixels. If PostScript Points are used if unit is missing. For instance, 8.27 in x 11.69 in, indicates the size of the paper in inches. Currently, the options can be single, which indicates that the control's content is exported to a single PDF page. For instance, the CopyTo("shot.pdf]33.11 in x 46.81 in[0 0 0 0]single") exports the control's content to an A0 single PDF page, with no margins.

• *.emf or any other extension determines the control to

save the control's content in **EMF** format.

For instance, the CopyTo("c:\temp\snapshot.png") property saves the control's content in PNG format to snapshot.png file.

Variant	A boolean expression that indicates whether the File was successful saved, if the File parameter is not empty, or a one dimension safe array of bytes, if the File parameter is empty string.

The CopyTo method copies/exports the control's view to BMP, PNG, JPG, GIF, TIFF, PDF or EMF graphical files, including no scroll bars. Use the <u>Copy</u> method to copy the control's content to the clipboard.

- The **BMP** file format, also known as bitmap image file or device independent bitmap (DIB) file format or simply a bitmap, is a raster graphics image file format used to store bitmap digital images, independently of the display device (such as a graphics adapter)
- The **JPEG** file format (seen most often with the .jpg extension) is a commonly used method of lossy compression for digital images, particularly for those images produced by digital photography.
- The **GIF** (Graphics Interchange Format) is a bitmap image format that was introduced by CompuServe in 1987 and has since come into widespread usage on the World Wide Web due to its wide support and portability.
- The **TIFF** (Tagged Image File Format) is a computer file format for storing raster graphics images, popular among graphic artists, the publishing industry, and both amateur and professional photographers in general.
- The **PNG** (Portable Network Graphics) is a raster graphics file format that supports lossless data compression. PNG was created as an improved, non-patented replacement for Graphics Interchange Format (GIF), and is the most used lossless image compression format on the Internet
- The **PDF** (Portable Document Format) is a file format used to present documents in a manner independent of application software, hardware, and operating systems. Each PDF file encapsulates a complete description of a fixed-layout flat document, including the text, fonts, graphics, and other information needed to display it.
- The EMF (Enhanced Metafile Format) is a 32-bit format that can contain both vector information and bitmap information. This format is an improvement over the Windows Metafile Format and contains extended features, such as the following

Built-in scaling information Built-in descriptions that are saved with the file Improvements in color palettes and device independence The EMF format is an extensible format, which means that a programmer can modify the original specification to add functionality or to meet specific needs. You can paste this format to Microsoft Word, Excel, Front Page, Microsoft Image Composer and any application that know to handle EMF formats.

The following VB sample saves the control's content to a EMF file:

If (Control.CopyTo("c:\temp\test.emf")) Then MsgBox "test.emf file created, open it using the mspaint editor." End If

The following VB sample prints the EMF content (as bytes, File parameter is empty string):

Dim i As Variant For Each i In Control.CopyTo("") Debug.Print i Next

property Schedule.CreateEventLabel as String

Specifies the label to be shown while creating events.

туре	Description
String	A String expression that defines the label to be displayed when the user creates new events. The CreateEventLabel supports extended HTML format as explained bellow.

By default, the CreateEventLabel property is: "<%=%256%>
<%=((1:=int(0:= 1/24/60/60/2*24))) != 0 ? =:1 + ' hour(s)' : ") + (=:1 ? ' ' : ") + ((1:=round((=:0 - =:1)*60))) $!= 0 ? = :1 + ' \min(s)' : ")$ %>" which displays the margins of the event being created on the first line, while on the second line it displays the duration of the newly event. The CreateEventLabel property indicates the HTML format to be shown on the label when the user creates a new event. The CreateEventLabelAlign property aligns the label being shown when the user creates a new event. The AllowCreateEvent property indicates the combination of keys that allows the user to create new events in the control. The AllowMoveEvent property indicates the combination of the keys to let user moves the events. The <u>AllowResizeEvent</u> property indicates the combination of the keys to let user resizes the events. The **Background**(exScheduleCreateEventBackColor) and Background(exScheduleCreateEventForeColor) specifies the visual appearance of the event being created. The AddEvent event occurs once a new event is added to the Events collection. The **DefaultEventPadding** property indicates the padding of the labels on the event, relative to event's borders.

Here's a few samples:

- "new", simple new text is shown.
- "<a no>title", displays a clickable text such as <u>title</u>, and <u>AnchorClick</u> can be used to determine whether the no anchor has been clicked.
- "<a>pic1:32", displays a click able image, the <u>AnchorClick</u> can be used to determine whether the anchor has been clicked. We would recommend using the <u>Pictures</u> or <u>ExtraPictures</u> property to assign pictures to an event.
- "<%=%256%>", displays the event's start and end points in a short format.
- "<%=%257%>", displays the event's margins in a long format.
- "Start: <%=%1%>
End: <%=%2%>", displays the starting margin of the even on the first line, while on the second line it displays the ending point of the event.
- "<%=%256%>
Caption: <%=%5%>", displays the event's margins in short format on the first line, and on the second line it displays the event's <u>Caption</u> property. The caption shown on the event's body is automatically updated once the event is moved to a new position or the event's Caption is changed.
- "<%=%256%>
<%=%264? `repetitive event`:``%>" displays automatically the

"repetitive event" for repetitive events, or when the event's <u>Repetitive</u> property is not empty and valid

- "Duration: <%=((1:=int(0:= (date(%2)-date(%1)))) != 0 ? (=:1 + ' day(s)') : ") + (=:1 ? ' ' : ") + ((1:=int(0:=((=:0 =:1 + 1/24/60/60/2)*24))) != 0 ? =:1 + ' hour(s)' : ") + (=:1 ? ' ' : ") + ((1:=round((=:0 =:1)*60)) != 0 ? =:1 + ' min(s)' : ")%>" displays the duration of the event in days, hours and minutes.
- "<%=%256%>
Duration: <%=((1:=int(0:= (date(%2)-date(%1)))) != 0 ? (=:1 + ' day(s)') : ") + (=:1 ? ' ' : ") + ((1:=int(0:=((=:0 =:1 + 1/24/60/60/2)*24))) != 0 ? =:1 + ' hour(s)' : ") + (=:1 ? ' ' : ") + ((1:=round((=:0 =:1)*60)) != 0 ? =:1 + ' min(s)' : ")%>" displays the event's margins on the first line and the duration of the event in days, hours and minutes, on the second line
- "<%=%><%=%5%>
<%=%256%>", displays the event's Caption on first line(s), following by the event's Start/End margins in short date-time format. The <%=%> prefix forces the expression to be re-evaluated and apply any HTML tag found. For instance, %5 indicates the event's Caption property, and if it contains HTML tags they will be applied as is, instead displaying them as a plain text. Any expression that starts with "<%=%>" is re- evaluated and its result is displayed in HTML format (available starting with the version 12.2)

The <u>EventKnowPropertyEnum</u> defines the %identifiers that can be used in formula <%=FORMULA%>. For instance, the CreateEventLabel property on "Start:<%=time(%1) replace `AM` with ``%>" displays the time when the event starts with no AM time indicators.

The property supports the following identifiers. These identifiers can be used in FORMULA format:

- %1, Indicates the starting date/time of the event as DATE type, equivalent with <u>Start</u> property
- %2, Indicates the ending date/time of the event as DATE type, equivalent with <u>End</u> property
- %3, Indicates if the current event is an all day event as BOOL type, equivalent with <u>AllDayEvent</u> property
- %4, Indicates the identifier of the event's group, as LONG type, equivalent with <u>GroupID</u> property.
- **%5**, Indicates the caption of the event, as STRING expression, equivalent with <u>Caption</u> property.
- %6, Indicates the extra data associated with the event, as VARIANT type, equivalent with <u>UserData</u> property.
- %7, Gets or sets the duration of the event as FLOAT expression. Above you can find how you can display the duration of the event in hours, minutes...
- %8, Specifies the repetitive expression of the event, equivalent with <u>Repetitive</u> property.

- %256, Gets the margins of the event in a short format, as a STRING expression. The <u>ShortDateFormat</u> property defines the short date format. The <u>ShortTimeFormat</u> property defines the short time format.
- %257, Gets the margins of the event in a long format, as a STRING expression. The <u>LongDateFormat</u> property defines the long date format. The <u>LongTimeFormat</u> property defines the long time format.
- %258, Gets the starting date (not including the time) of the current event, as a DATE type.
- %259, Gets the starting time (not including the date) of the current event, as DATE type from 0 to 1.
- %260, Gets the ending date (not including the time) of the current event, as a DATE type.
- %261, Gets the ending time (not including the date) of the current event, as DATE type from 0 to 1.
- %262, Gets the label of the owner group, as STRING expression, equivalent with <u>Caption</u> property of the Group's event.
- %263, Gets the title of the owner group, as STRING expression, equivalent with <u>Title</u> property of the Group's event.
- %264, Indicates if the current event is a repetitive event, as BOOL type. This flag returns TRUE, if the <u>Repetitive</u> property is not empty, and the expression is valid.

The FORMULA, is identified by <%=FORMULA%>, and supports 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#, #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
 - $\circ\,$ #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
- Itrim (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 property supports the following built-in HTML tags:

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

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

 exp, stores the plain text to be shown once the user clicks the anchor, such as "<a ;exp=show lines>"

 $\circ~$ e64, encodes in BASE64 the HTML text to be shown once the user clicks the anchor, such as "anchor, such as "anchor, such as "anchor, such as

;e64=gA8ABmABnABjABvABshIAOQAEAAHAAGESikWio+ABzABohp3iELABpABu " that displays show lines- in gray when the user clicks the + anchor. The "gA8ABmABnABjABvABshIAOQAEAAHAAGESikWio+ABzABohp3iELABpABuABljY string encodes the "<fgcolor 808080>show lines<a>-</fgcolor>" The Decode64Text/Encode64Text methods of the eXPrint can be used to decode/encode e64 fields.

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

underlined and bold on the first line and Line1, Line2, Line3 on the rest. The "show lines" is shown instead of Line1, Line2, Line3 once the user clicks the + sign.

- ... displays portions of text with a different font and/or different size. For instance, the "bit" draws the bit text using the Tahoma font, on size 12 pt. If the name of the font is missing, and instead size is present, the current font is used with a different size. For instance, "bit" displays the bit text using the current font, but with a different size.
- **<fgcolor rrggbb>** ... **</fgcolor>** or **<**fgcolor=rrggbb> ... **</**fgcolor> displays text with a specified foreground color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- <bgcolor rrggbb> ... </bgcolor> or <bgcolor=rrggbb> ... </bgcolor> displays text with a specified background color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- <solidline rrggbb> ... </solidline> or <solidline=rrggbb> ... </solidline> draws a solidline 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
- number[:width] 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.
- key[:width] 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 FFFFF;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" generates the following picture:

shadow

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

outline anti-aliasing

property Schedule.CreateEventLabelAlign as ContentAlignmentEnum

Specifies the alignment of the label to be shown while creating events.

Туре	Description
<u>ContentAlignmentEnum</u>	A ContentAlignmentEnum expression that specifies the alignment of the label being shown when the user creates new events.

The CreateEventLabelAlign property aligns the label being shown when the user creates a new event. The <u>CreateEventLabel</u> property indicates the HTML format to be shown on the label when the user creates a new event. The <u>UpdateEventsLabelAlign</u> property aligns the label being shown when the user moves or resizes the events.

property Schedule.DataField(Property as EventKnownPropertyEnum) as Variant

Automatically updates / synchronizes the known property of the event with the associated data field and reverse.

Туре	Description
Property as <u>EventKnownPropertyEnum</u>	A Property to be associated with a field in the record set
Variant	A String expression that specifies the name of the field to be associated with the Property. A Long expression that specifies the index of the field in the recordset.

By default, the DataField(exEventStartDateTime) property is "Start" and the DataField(exEventEndDateTime) is "End". Use the DataField property to associate the exEventStartDateTime and exEventEndDateTime or exEventDuration with the fields in your data source. The DataField property must be called before calling the <u>DataSource</u> property. The <u>DataSource</u> property binds a recordset to your control. Use the DataField property to associate a <u>property</u> of the event with a field in the database. This way when the property is changed it is updated in the associated field. The control fires the <u>Error</u> event if any error occurs when handling the data source. The <u>Syncronize</u> method ensures that each record has associated an event, and each event has associated a record.

The following samples shows how you can bound the control to a data source:

VBA

```
' Error event - Fired when an internal error occurs.
Private Sub Schedule1_Error(ByVal Error As Long,ByVal Description As String)
With Schedule1
Debug.Print( Description )
End With
End Sub
With Schedule1
Set rs = CreateObject("ADOR.Recordset")
With rs
.Open "Events", "Provider=Microsoft.ACE.OLEDB.12.0;Data Source=C:\Program
Files\Exontrol\ExSchedule\Sample\Access2007\datasource.accdb",3,3
End With
.BeginUpdate
```

```
.Calendar.Selection = #11/11/2013#

.DataField(1) = "Start"

.DataField(2) = "End"

.DataField(11) = "Extra"

.DataSource = rs

.EndUpdate

End With
```

VB6

```
' Error event - Fired when an internal error occurs.
Private Sub Schedule1_Error(ByVal Error As Long, ByVal Description As String)
  With Schedule1
    Debug.Print(Description)
  End With
End Sub
With Schedule1
  Set rs = CreateObject("ADOR.Recordset")
  With rs
    .Open "Events", "Provider=Microsoft.ACE.OLEDB.12.0; Data Source=C:\Program
Files\Exontrol\ExSchedule\Sample\Access2007\datasource.accdb",3,3
  End With
  .BeginUpdate
  .Calendar.Selection = #11/11/2013#
  .DataField(exEventStartDateTime) = "Start"
  .DataField(exEventEndDateTime) = "End"
  .DataField(exEventExtraLabel) = "Extra"
  .DataSource = rs
  .EndUpdate
End With
```

VB.NET

' Error event - Fired when an internal error occurs.

Private Sub Exschedule1_Error(ByVal sender As System.Object,ByVal Err As Integer,ByVal Description As String) Handles Exschedule1.Error With Exschedule1

```
Debug.Print( Description )
End With
End Sub
Dim rs
With Exschedule1
rs = New ADODB.Recordset()
With rs
.Open("Events","Provider=Microsoft.ACE.OLEDB.12.0;Data Source=C:\Program
Files\Exontrol\ExSchedule\Sample\Access2007\datasource.accdb",3,3)
End With
.BeginUpdate()
.Calendar.Selection = #11/11/2013#
```

.set_DataField(exontrol.EXSCHEDULELib.EventKnownPropertyEnum.exEventStartDateTime,

.set_DataField(exontrol.EXSCHEDULELib.EventKnownPropertyEnum.exEventEndDateTime,"

.set_DataField(exontrol.EXSCHEDULELib.EventKnownPropertyEnum.exEventExtraLabel,"Ext

.**DataSource** = rs .EndUpdate() End With

VB.NET for /COM

```
    'Error event - Fired when an internal error occurs.
    Private Sub AxSchedule1_Error(ByVal sender As System.Object, ByVal e As
    AxEXSCHEDULELib._IScheduleEvents_ErrorEvent) Handles AxSchedule1.Error
    With AxSchedule1
    Debug.Print( e.description )
    End With
    End Sub
```

Dim rs

```
With AxSchedule1
rs = CreateObject("ADOR.Recordset")
With rs
.Open("Events","Provider=Microsoft.ACE.OLEDB.12.0;Data Source=C:\Program
Files\Exontrol\ExSchedule\Sample\Access2007\datasource.accdb",3,3)
End With
.BeginUpdate()
.Calendar.Selection = #11/11/2013#
```

.set_DataField(EXSCHEDULELib.EventKnownPropertyEnum.exEventStartDateTime,"Start")
.set_DataField(EXSCHEDULELib.EventKnownPropertyEnum.exEventEndDateTime,"End")
.set_DataField(EXSCHEDULELib.EventKnownPropertyEnum.exEventExtraLabel,"Extra")
.DataSource = rs
.EndUpdate()
End With

C++

```
// Error event - Fired when an internal error occurs.
void OnErrorSchedule1(long Error,LPCTSTR Description)
{
  /*
    Copy and paste the following directives to your header file as
    it defines the namespace 'EXSCHEDULELib' for the library: 'ExSchedule 1.0 Control
Library'
    #import <ExSchedule.dll>
    using namespace EXSCHEDULELib;
  */
  EXSCHEDULELib::ISchedulePtr spSchedule1 = GetDlgItem(IDC_SCHEDULE1)-
>GetControlUnknown();
  OutputDebugStringW(L"Description");
}
EXSCHEDULELib::ISchedulePtr spSchedule1 = GetDlgItem(IDC_SCHEDULE1)-
>GetControlUnknown();
/*
  Includes the definition for CreateObject function like follows:
```

```
#include <comdef.h>
  IUnknownPtr CreateObject( BSTR Object )
    IUnknownPtr spResult;
    spResult.CreateInstance(Object);
    return spResult;
  };
*/
  Copy and paste the following directives to your header file as
  it defines the namespace 'ADODB' for the library: 'Microsoft ActiveX Data Objects 6.0
Library'
  #import <msado15.dll> rename("EOF","REOF")
ADODB::_RecordsetPtr rs = ::CreateObject(L"ADOR.Recordset");
  rs->Open("Events",_bstr_t("Provider=Microsoft.ACE.OLEDB.12.0;Data
Source=C:\\Program
Files\\Exontrol\\ExSchedule\\Sample\\Access2007\\datasource.accdb",ADODB::adOpenSta
spSchedule1->BeginUpdate();
spSchedule1->GetCalendar()->PutSelection("11/11/2013");
spSchedule1->PutDataField(EXSCHEDULELib::exEventStartDateTime,"Start");
spSchedule1->PutDataField(EXSCHEDULELib::exEventEndDateTime,"End");
spSchedule1->PutDataField(EXSCHEDULELib::exEventExtraLabel,"Extra");
spSchedule1->PutDataSource(((ADODB::_RecordsetPtr)(rs)));
spSchedule1->EndUpdate();
```

C++ Builder

```
// Error event - Fired when an internal error occurs.
void __fastcall TForm1::Schedule1Error(TObject *Sender,long Error,BSTR Description)
{
    OutputDebugString( L"Description" );
}
```

Select the Component\Import Component...\Import a Type Library, to import the following Type Library:

Microsoft ActiveX Data Objects 6.0 Library

TypeLib: C:\Program Files\Common Files\System\ado\msado15.dll

to define the namespace: Adodb_tlb

//#include "ADODB_TLB.h"

```
Adodb_tlb::_RecordsetPtr rs = Variant::CreateObject(L"ADOR.Recordset");
    rs->Open(TVariant("Events"),TVariant(String("Provider=Microsoft.ACE.OLEDB.12.0;Data
    Source=C:\\Program
Files\\Exontrol\\ExSchedule\\Sample\\Access2007\\datasource.accdb"),Adodb_tlb::CursorT
```

```
Schedule1->BeginUpdate();
Schedule1->Calendar->set_Selection(TVariant(TDateTime(2013,11,11).operator
double()));
Schedule1-
>DataField[Exschedulelib_tlb::EventKnownPropertyEnum::exEventStartDateTime] =
TVariant("Start");
Schedule1-
>DataField[Exschedulelib_tlb::EventKnownPropertyEnum::exEventEndDateTime] =
TVariant("End");
Schedule1->DataField[Exschedulelib_tlb::EventKnownPropertyEnum::exEventExtraLabel]
= TVariant("Extra");
Schedule1->DataField[Exschedulelib_tlb::EventKnownPropertyEnum::exEventExtraLabel]
= TVariant("Extra");
Schedule1->DataSource = (IDispatch*)rs;
Schedule1->EndUpdate();
```

C#

*/

// Error event - Fired when an internal error occurs.

private void exschedule1_Error(object sender,int Err,string Description)

{

System.Diagnostics.Debug.Print(Description.ToString());

//this.exschedule1.Error += new exontrol.EXSCHEDULELib.exg2antt.ErrorEventHandler(this.exschedule1_Error);

// Add 'Microsoft ActiveX Data Objects 6.0 Library' reference to your project.

ADODB.Recordset rs = new ADODB.Recordset();

rs.Open("Events","Provider=Microsoft.ACE.OLEDB.12.0;Data Source=C:\\Program Files\\Exontrol\\ExSchedule\\Sample\\Access2007\\datasource.accdb",exontrol.ADODB.Cu

exschedule1.BeginUpdate();

exschedule1.Calendar.Selection =

Convert.ToDateTime("11/11/2013",System.Globalization.CultureInfo.GetCultureInfo("en-US"));

exschedule1.set_DataField(exontrol.EXSCHEDULELib.EventKnownPropertyEnum.exEventSta

exschedule1.set_DataField(exontrol.EXSCHEDULELib.EventKnownPropertyEnum.exEventEn

exschedule1.set_DataField(exontrol.EXSCHEDULELib.EventKnownPropertyEnum.exEventEx

```
exschedule1.DataSource = (rs as ADODB.Recordset);
exschedule1.EndUpdate();
```

JavaScript

```
<SCRIPT FOR="Schedule1" EVENT="Error(Error,Description)" LANGUAGE="JScript">
alert( Description );
</SCRIPT>
```

```
<OBJECT classid = "clsid:9B09E13D-7A88-4299-9DBE-383380435377" id = "Schedule1" > </OBJECT >
```

```
<SCRIPT LANGUAGE="JScript">
var rs = new ActiveXObject("ADOR.Recordset");
rs.Open("Events","Provider=Microsoft.ACE.OLEDB.12.0;Data Source=C:\\Program
Files\\Exontrol\\ExSchedule\\Sample\\Access2007\\datasource.accdb",3,3,null);
Schedule1.BeginUpdate();
Schedule1.Calendar.Selection = "11/11/2013";
```

```
Schedule1.DataField(1) = "Start";
Schedule1.DataField(2) = "End";
Schedule1.DataField(11) = "Extra";
Schedule1.DataSource = rs;
Schedule1.EndUpdate();
</SCRIPT>
```

C# for /COM

```
// Error event - Fired when an internal error occurs.
private void axSchedule1_Error(object sender,
AxEXSCHEDULELib._IScheduleEvents_ErrorEvent e)
{
  System.Diagnostics.Debug.Print( e.description.ToString() );
}
//this.axSchedule1.Error += new
AxEXSCHEDULELib. IScheduleEvents_ErrorEventHandler(this.axSchedule1_Error);
// Add 'Microsoft ActiveX Data Objects 6.0 Library' reference to your project.
ADODB.Recordset rs = new ADODB.Recordset();
  rs.Open("Events", "Provider=Microsoft.ACE.OLEDB.12.0; Data Source=C:\\Program
Files\\Exontrol\\ExSchedule\\Sample\\Access2007\\datasource.accdb",ADODB.CursorTypel
axSchedule1.BeginUpdate();
axSchedule1.Calendar.Selection =
Convert.ToDateTime("11/11/2013",System.Globalization.CultureInfo.GetCultureInfo("en-
US"));
axSchedule1.set_DataField(EXSCHEDULELib.EventKnownPropertyEnum.exEventStartDateT
axSchedule1.set_DataField(EXSCHEDULELib.EventKnownPropertyEnum.exEventEndDateTir
axSchedule1.set_DataField(EXSCHEDULELib.EventKnownPropertyEnum.exEventExtraLabel,
```

```
axSchedule1.DataSource = (rs as ADODB.Recordset);
axSchedule1.EndUpdate();
```

X++ (Dynamics Ax 2009)

```
// Error event - Fired when an internal error occurs.
void onEvent_Error(int _Error,str _Description)
  print(_Description);
public void init()
  anytype rs;
  str var_s;
  super();
  // Add 'Microsoft ActiveX Data Objects 6.0 Library' reference to your project.
  rs = COM::createFromObject(new ADODB.Recordset()); rs = rs;
    var_s = "Provider=Microsoft.ACE.OLEDB.12.0;Data Source=C:\\Program
Files\\Exontrol\\ExSchedule\\Sample\\Access2007\\datasource.accdb";
rs.Open("Events",COMVariant::createFromStr(var_s),3/*adOpenStatic*/,3/*adLockOptimistic
  exschedule1.BeginUpdate();
exschedule1.Calendar().Selection(COMVariant::createFromDate(str2Date("11/11/2013",213)
```

```
exschedule1.DataField(1/*exEventStartDateTime*/,"Start");
exschedule1.DataField(2/*exEventEndDateTime*/,"End");
exschedule1.DataField(11/*exEventExtraLabel*/,"Extra");
exschedule1.DataSource(rs);
exschedule1.EndUpdate();
```

VFP

}

```
*** Error event - Fired when an internal error occurs. ***
LPARAMETERS Error, Description
```

```
with thisform.Schedule1
    DEBUGOUT(Description)
  endwith
with thisform.Schedule1
  rs = CreateObject("ADOR.Recordset")
  with rs
    .Open("Events", "Provider=Microsoft.ACE.OLEDB.12.0; Data Source=C:\Program
Files\Exontrol\ExSchedule\Sample\Access2007\datasource.accdb",3,3)
  endwith
  .BeginUpdate
  .Calendar.Selection = {^2013-11-11}
  .Object.DataField(1) = "Start"
  .Object.DataField(2) = "End"
  .Object.DataField(11) = "Extra"
  .DataSource = rs
  .EndUpdate
endwith
```

dBASE Plus

```
/*
with (this.ACTIVEX1.nativeObject)
Error = class::nativeObject_Error
endwith
*/
// Fired when an internal error occurs.
function nativeObject_Error(Error,Description)
local oSchedule
oSchedule = form.Activex1.nativeObject
? Str(Description)
return
local oSchedule,rs
oSchedule = form Activex1 pativeObject
```

```
oSchedule = form.Activex1.nativeObject
rs = new OleAutoClient("ADOR.Recordset")
```

```
rs.Open("Events","Provider=Microsoft.ACE.OLEDB.12.0;Data Source=C:\Program
Files\Exontrol\ExSchedule\Sample\Access2007\datasource.accdb",3,3)
oSchedule.BeginUpdate()
oSchedule.Calendar.Selection = "11/11/2013"
oSchedule.Template = [DataField(1) = "Start"] // oSchedule.DataField(1) = "Start"
oSchedule.Template = [DataField(2) = "End"] // oSchedule.DataField(2) = "End"
oSchedule.Template = [DataField(11) = "Extra"] // oSchedule.DataField(11) = "Extra"
oSchedule.Template = [DataField(11) = "Extra"] // oSchedule.DataField(11) = "Extra"
```

XBasic (Alpha Five)

```
' Fired when an internal error occurs.
function Error as v (Error as N, Description as C)
  Dim oSchedule as P
  oSchedule = topparent:CONTROL_ACTIVEX1.activex
  ? Description
end function
Dim oSchedule as P
Dim rs as P
oSchedule = topparent:CONTROL_ACTIVEX1.activex
rs = OLE.Create("ADOR.Recordset")
  rs.Open("Events", "Provider=Microsoft.ACE.OLEDB.12.0; Data Source=C:\Program
Files\Exontrol\ExSchedule\Sample\Access2007\datasource.accdb",3,3)
oSchedule.BeginUpdate()
oSchedule.Calendar.Selection = {11/11/2013}
oSchedule.Template = "DataField(1) = \"Start\"" ' oSchedule.DataField(1) = "Start"
oSchedule.Template = "DataField(2) = \"End\"" ' oSchedule.DataField(2) = "End"
oSchedule.Template = "DataField(11) = \"Extra\"" ' oSchedule.DataField(11) = "Extra"
oSchedule.DataSource = rs
oSchedule.EndUpdate()
```

Delphi 8 (.NET only)

// Error event - Fired when an internal error occurs. procedure TWinForm1.AxSchedule1_Error(sender: System.Object; e:

```
AxEXSCHEDULELib._IScheduleEvents_ErrorEvent);
begin
  with AxSchedule1 do
  begin
    OutputDebugString( e.description );
  end
end;
with AxSchedule1 do
begin
  rs := (ComObj.CreateComObject(ComObj.ProgIDToClassID('ADOR.Recordset')) as
ADODB.Recordset);
  with rs do
  begin
    Open('Events','Provider=Microsoft.ACE.OLEDB.12.0;Data Source=C:\Program
Files\Exontrol\ExSchedule\Sample\Access2007\datasource.accdb',3,3,Nil);
  end;
  BeginUpdate();
  Calendar.Selection := '11/11/2013';
```

set_DataField(EXSCHEDULELib.EventKnownPropertyEnum.exEventStartDateTime,'Start');
set_DataField(EXSCHEDULELib.EventKnownPropertyEnum.exEventEndDateTime,'End');
set_DataField(EXSCHEDULELib.EventKnownPropertyEnum.exEventExtraLabel,'Extra');
DataSource := (rs as ADODB.Recordset);
EndUpdate();
end

Delphi (standard)

```
// Error event - Fired when an internal error occurs.
procedure TForm1.Schedule1Error(ASender: TObject; Error : Integer;Description :
WideString);
begin
with Schedule1 do
begin
OutputDebugString( Description );
end
```

end;

```
with Schedule1 do
```

begin

rs :=

```
(IUnknown(ComObj.CreateComObject(ComObj.ProgIDToClassID('ADOR.Recordset'))) as ADODB_TLB.Recordset);
```

with rs do

begin

```
Open('Events','Provider=Microsoft.ACE.OLEDB.12.0;Data Source=C:\Program
Files\Exontrol\ExSchedule\Sample\Access2007\datasource.accdb',3,3,Null);
```

end;

```
BeginUpdate();
Calendar.Selection := '11/11/2013';
DataField[EXSCHEDULELib_TLB.exEventStartDateTime] := 'Start';
DataField[EXSCHEDULELib_TLB.exEventEndDateTime] := 'End';
DataField[EXSCHEDULELib_TLB.exEventExtraLabel] := 'Extra';
DataSource := (IUnknown(rs) as ADODB_TLB.Recordset);
EndUpdate();
end
```

Visual Objects

```
METHOD OCX_Exontrol1Error(Error,Description) CLASS MainDialog

// Error event - Fired when an internal error occurs.

OutputDebugString(String2Psz(AsString(Description)))

RETURN NIL
```

local rs as _Recordset

// Generate Source for 'Microsoft ActiveX Data Objects 6.0 Library' server from Tools\Automation Server...

```
rs := _Recordset{"ADOR.Recordset"}
rs:Open("Events","Provider=Microsoft.ACE.OLEDB.12.0;Data Source=C:\Program
Files\Exontrol\ExSchedule\Sample\Access2007\datasource.accdb",3,3,0)
oDCOCX_Exontrol1:BeginUpdate()
oDCOCX_Exontrol1:Calendar:Selection := SToD("20131111")
```

oDCOCX_Exontrol1:[DataField,exEventStartDateTime] := "Start" oDCOCX_Exontrol1:[DataField,exEventEndDateTime] := "End" oDCOCX_Exontrol1:[DataField,exEventExtraLabel] := "Extra" oDCOCX_Exontrol1:DataSource := _Recordset{rs} oDCOCX_Exontrol1:EndUpdate()

PowerBuilder

```
/*begin event Error(long Error, string Description) - Fired when an internal error occurs.*/
  OleObject oSchedule
  oSchedule = ole_1.Object
  MessageBox("Information", string(String(Description)))
*/
/*end event Error*/
OleObject oSchedule,rs
oSchedule = ole_1.Object
rs = CREATE OLEObject
rs.ConnectToNewObject("ADOR.Recordset")
  rs.Open("Events", "Provider=Microsoft.ACE.OLEDB.12.0;Data Source=C:\Program
Files\Exontrol\ExSchedule\Sample\Access2007\datasource.accdb",3,3)
oSchedule.BeginUpdate()
oSchedule.Calendar.Selection = 2013-11-11
oSchedule.DataField(1,"Start")
oSchedule.DataField(2,"End")
oSchedule.DataField(11,"Extra")
oSchedule.DataSource = rs
oSchedule.EndUpdate()
```

Visual DataFlex

// Fired when an internal error occurs.

Procedure OnComError Integer IIError String IIDescription Forward Send OnComError IIError IIDescription ShowIn IIDescription End_Procedure Procedure OnCreate Forward Send OnCreate Variant rs Get Comcreateobject "ADOR.Recordset" to rs Send ComOpen "Events" "Provider=Microsoft.ACE.OLEDB.12.0;Data Source=C:\Program Files\Exontrol\ExSchedule\Sample\Access2007\datasource.accdb" OLEadOpenStatic OLEadLockOptimistic Nothing Send ComBeginUpdate Variant voCalendar Get ComCalendar to voCalendar Handle hoCalendar Get Create (RefClass(cComCalendar)) to hoCalendar Set pvComObject of hoCalendar to voCalendar Set ComSelection of hoCalendar to "11/11/2013" Send Destroy to hoCalendar Set ComDataField OLEexEventStartDateTime to "Start" Set **ComDataField** OLEexEventEndDateTime to "End" Set **ComDataField** OLEexEventExtraLabel to "Extra" Set ComDataSource to rs Send ComEndUpdate End_Procedure

Xbase++

```
PROCEDURE OnError(oSchedule,Error,Description)
    DevOut( Transform(Description,"") )
RETURN
#include "AppEvent.ch"
#include "ActiveX.ch"
```

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

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

```
oSchedule := XbpActiveXControl():new( oForm:drawingArea )
    oSchedule:CLSID := "Exontrol.Schedule.1" /*{9B09E13D-7A88-4299-9DBE-
383380435377}*/
    oSchedule:create(,, {10,60},{610,370} )
```

```
oSchedule:Error := {|Error,Description| OnError(oSchedule,Error,Description)} /*Fired
```

when an internal error occurs.*/

```
rs := CreateObject("ADOR.Recordset")
rs:Open("Events","Provider=Microsoft.ACE.OLEDB.12.0;Data Source=C:\Program
Files\Exontrol\ExSchedule\Sample\Access2007\datasource.accdb",3/*adOpenStatic*/,3/*ad
```

```
oSchedule:BeginUpdate()
oSchedule:Calendar():Selection := "11/11/2013"
oSchedule:SetProperty("DataField",1/*exEventStartDateTime*/,"Start")
oSchedule:SetProperty("DataField",2/*exEventEndDateTime*/,"End")
oSchedule:SetProperty("DataField",11/*exEventExtraLabel*/,"Extra")
oSchedule:DataSource := rs
oSchedule:EndUpdate()
```

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

property Schedule.DataSource as Object

Retrieves or sets a value that indicates the data source for object.

Туре	Description
Object	An Object that defines the control's data. Currently, the control accepts ADO.Recordset, ADODB.Recordset objects, DAO recordsets

The DataSource property binds the control to an ADO.Recordset, ADODB.Recordset objects, DAO recordsets. *Before calling the DataSource property, the DataField property must associate the start/end of the events to fields in the database.* In other words, the DataField(exEventStartDateTime) and DataField(exEventEndDateTime) or DataField(exEventDuration) must be associated with fields in the database. Use the DataField property to associate a property of the event with a field in the database. This way when the property is changed it is updated in the associated field. The control fires the Error event if any error occurs when handling the data source. The Syncronize method ensures that each record has associated an event, and each event has associated a record.

The following samples shows how you can bound the control to a data source:

VBA

```
' Error event - Fired when an internal error occurs.
Private Sub Schedule1_Error(ByVal Error As Long, ByVal Description As String)
  With Schedule1
    Debug.Print( Description )
  End With
End Sub
With Schedule1
  Set rs = CreateObject("ADOR.Recordset")
  With rs
    .Open "Events", "Provider=Microsoft.ACE.OLEDB.12.0; Data Source=C:\Program
Files\Exontrol\ExSchedule\Sample\Access2007\datasource.accdb",3,3
  End With
  .BeginUpdate
  .Calendar.Selection = #11/11/2013#
  .DataField(1) = "Start"
  .DataField(2) = "End"
```

```
.DataField(11) = "Extra"
.DataSource = rs
.EndUpdate
```

VB6

End With

```
' Error event - Fired when an internal error occurs.
Private Sub Schedule1_Error(ByVal Error As Long, ByVal Description As String)
  With Schedule1
    Debug.Print( Description )
  End With
End Sub
With Schedule1
  Set rs = CreateObject("ADOR.Recordset")
  With rs
    .Open "Events", "Provider=Microsoft.ACE.OLEDB.12.0; Data Source=C:\Program
Files\Exontrol\ExSchedule\Sample\Access2007\datasource.accdb",3,3
  End With
  .BeginUpdate
  .Calendar.Selection = #11/11/2013#
  .DataField(exEventStartDateTime) = "Start"
  .DataField(exEventEndDateTime) = "End"
  .DataField(exEventExtraLabel) = "Extra"
  .DataSource = rs
  .EndUpdate
End With
```

VB.NET

'Error event - Fired when an internal error occurs.
Private Sub Exschedule1_Error(ByVal sender As System.Object,ByVal Err As Integer,ByVal Description As String) Handles Exschedule1.Error
With Exschedule1
Debug.Print(Description)
End With
End Sub

```
Dim rs
With Exschedule1
rs = New ADODB.Recordset()
With rs
.Open("Events","Provider=Microsoft.ACE.OLEDB.12.0;Data Source=C:\Program
Files\Exontrol\ExSchedule\Sample\Access2007\datasource.accdb",3,3)
End With
.BeginUpdate()
.Calendar.Selection = #11/11/2013#
```

.set_DataField(exontrol.EXSCHEDULELib.EventKnownPropertyEnum.exEventStartDateTime,

.set_DataField(exontrol.EXSCHEDULELib.EventKnownPropertyEnum.exEventEndDateTime,"

.set_DataField(exontrol.EXSCHEDULELib.EventKnownPropertyEnum.exEventExtraLabel,"Ext

```
.DataSource = rs
.EndUpdate()
End With
```

VB.NET for /COM

```
' Error event - Fired when an internal error occurs.
Private Sub AxSchedule1_Error(ByVal sender As System.Object, ByVal e As
AxEXSCHEDULELib._IScheduleEvents_ErrorEvent) Handles AxSchedule1.Error
With AxSchedule1
Debug.Print( e.description )
End With
End Sub
Dim rs
With AxSchedule1
rs = CreateObject("ADOR.Recordset")
With rs
```

```
.Open("Events","Provider=Microsoft.ACE.OLEDB.12.0;Data Source=C:\Program
Files\Exontrol\ExSchedule\Sample\Access2007\datasource.accdb",3,3)
End With
.BeginUpdate()
```

```
.Calendar.Selection = #11/11/2013#
```

IUnknownPtr CreateObject(BSTR Object)

.set_DataField(EXSCHEDULELib.EventKnownPropertyEnum.exEventStartDateTime,"Start")
.set_DataField(EXSCHEDULELib.EventKnownPropertyEnum.exEventEndDateTime,"End")
.set_DataField(EXSCHEDULELib.EventKnownPropertyEnum.exEventExtraLabel,"Extra")
.DataSource = rs
.EndUpdate()
End With

C++

```
// Error event - Fired when an internal error occurs.
void OnErrorSchedule1(long Error,LPCTSTR Description)
{
  /*
    Copy and paste the following directives to your header file as
    it defines the namespace 'EXSCHEDULELib' for the library: 'ExSchedule 1.0 Control
Library'
    #import <ExSchedule.dll>
    using namespace EXSCHEDULELib;
  */
  EXSCHEDULELib::ISchedulePtr spSchedule1 = GetDlgItem(IDC_SCHEDULE1)-
>GetControlUnknown();
  OutputDebugStringW( L"Description" );
}
EXSCHEDULELib::ISchedulePtr spSchedule1 = GetDlgItem(IDC_SCHEDULE1)-
>GetControlUnknown();
/*
  Includes the definition for CreateObject function like follows:
  #include <comdef.h>
```

```
IUnknownPtr spResult;
    spResult.CreateInstance(Object);
    return spResult;
  };
*/
  Copy and paste the following directives to your header file as
  it defines the namespace 'ADODB' for the library: 'Microsoft ActiveX Data Objects 6.0
Library'
  #import <msado15.dll> rename("EOF","REOF")
*/
ADODB::_RecordsetPtr rs = ::CreateObject(L"ADOR.Recordset");
  rs->Open("Events",_bstr_t("Provider=Microsoft.ACE.OLEDB.12.0;Data
Source=C:\\Program
Files\\Exontrol\\ExSchedule\\Sample\\Access2007\\datasource.accdb",ADODB::adOpenSta
spSchedule1->BeginUpdate();
```

```
spSchedule1->PutDataField(EXSCHEDULELib::exEventStartDateTime,"Start");
spSchedule1->PutDataField(EXSCHEDULELib::exEventEndDateTime,"End");
spSchedule1->PutDataField(EXSCHEDULELib::exEventExtraLabel,"Extra");
spSchedule1->PutDataSource(((ADODB::_RecordsetPtr)(rs)));
spSchedule1->EndUpdate();
```

spSchedule1->GetCalendar()->PutSelection("11/11/2013");

C++ Builder

```
// Error event - Fired when an internal error occurs.
void __fastcall TForm1::Schedule1Error(TObject *Sender,long Error,BSTR Description)
{
    OutputDebugString( L"Description" );
}
/*
    Select the Component\Import Component...\Import a Type Library,
```

to import the following Type Library:

Microsoft ActiveX Data Objects 6.0 Library

TypeLib: C:\Program Files\Common Files\System\ado\msado15.dll

to define the namespace: Adodb_tlb

//#include "ADODB_TLB.h"

```
Adodb_tlb::_RecordsetPtr rs = Variant::CreateObject(L"ADOR.Recordset");
    rs->Open(TVariant("Events"),TVariant(String("Provider=Microsoft.ACE.OLEDB.12.0;Data
    Source=C:\\Program
Files\\Exontrol\\ExSchedule\\Sample\\Access2007\\datasource.accdb"),Adodb_tlb::CursorT
```

Schedule1->BeginUpdate(); Schedule1->Calendar->set_Selection(TVariant(TDateTime(2013,11,11).operator double())); Schedule1->**DataField**[Exschedulelib_tlb::EventKnownPropertyEnum::exEventStartDateTime] = TVariant("Start"); Schedule1->**DataField**[Exschedulelib_tlb::EventKnownPropertyEnum::exEventEndDateTime] = TVariant("End"); Schedule1->**DataField**[Exschedulelib_tlb::EventKnownPropertyEnum::exEventExtraLabel] = TVariant("Extra"); Schedule1->**DataField**[Exschedulelib_tlb::EventKnownPropertyEnum::exEventExtraLabel] = TVariant("Extra");

Schedule1->EndUpdate();

C#

*/

```
// Error event - Fired when an internal error occurs.
private void exschedule1_Error(object sender,int Err,string Description)
{
    System.Diagnostics.Debug.Print( Description.ToString() );
}
//this.exschedule1.Error += new
```

exontrol.EXSCHEDULELib.exg2antt.ErrorEventHandler(this.exschedule1_Error);

// Add 'Microsoft ActiveX Data Objects 6.0 Library' reference to your project.

ADODB.Recordset rs = new ADODB.Recordset();

rs.Open("Events","Provider=Microsoft.ACE.OLEDB.12.0;Data Source=C:\\Program Files\\Exontrol\\ExSchedule\\Sample\\Access2007\\datasource.accdb",exontrol.ADODB.Cu

exschedule1.BeginUpdate();

exschedule1.Calendar.Selection =

Convert.ToDateTime("11/11/2013",System.Globalization.CultureInfo.GetCultureInfo("en-US"));

exschedule1.set_DataField(exontrol.EXSCHEDULELib.EventKnownPropertyEnum.exEventSta

exschedule1.set_DataField(exontrol.EXSCHEDULELib.EventKnownPropertyEnum.exEventEn

exschedule1.set_DataField(exontrol.EXSCHEDULELib.EventKnownPropertyEnum.exEventEx

```
exschedule1.DataSource = (rs as ADODB.Recordset);
exschedule1.EndUpdate();
```

JavaScript

```
<SCRIPT FOR="Schedule1" EVENT="Error(Error,Description)" LANGUAGE="JScript">
alert( Description );
</SCRIPT>
```

```
<OBJECT classid = "clsid:9B09E13D-7A88-4299-9DBE-383380435377" id = "Schedule1" > </OBJECT >
```

<SCRIPT LANGUAGE="JScript"> var rs = new ActiveXObject("ADOR.Recordset"); rs.Open("Events","Provider=Microsoft.ACE.OLEDB.12.0;Data Source=C:\\Program Files\\Exontrol\\ExSchedule\\Sample\\Access2007\\datasource.accdb",3,3,null); Schedule1.BeginUpdate(); Schedule1.Calendar.Selection = "11/11/2013"; Schedule1.DataField(1) = "Start"; Schedule1.DataField(2) = "End"; Schedule1.DataField(11) = "Extra";

```
Schedule1.DataSource = rs;
Schedule1.EndUpdate();
</SCRIPT>
```

C# for /COM

```
// Error event - Fired when an internal error occurs.
private void axSchedule1_Error(object sender,
AxEXSCHEDULELib._IScheduleEvents_ErrorEvent e)
{
  System.Diagnostics.Debug.Print( e.description.ToString() );
}
//this.axSchedule1.Error += new
AxEXSCHEDULELib. IScheduleEvents_ErrorEventHandler(this.axSchedule1_Error);
// Add 'Microsoft ActiveX Data Objects 6.0 Library' reference to your project.
ADODB.Recordset rs = new ADODB.Recordset();
  rs.Open("Events","Provider=Microsoft.ACE.OLEDB.12.0;Data Source=C:\\Program
Files\\Exontrol\\ExSchedule\\Sample\\Access2007\\datasource.accdb",ADODB.CursorTypel
axSchedule1.BeginUpdate();
axSchedule1.Calendar.Selection =
Convert.ToDateTime("11/11/2013",System.Globalization.CultureInfo.GetCultureInfo("en-
US"));
axSchedule1.set_DataField(EXSCHEDULELib.EventKnownPropertyEnum.exEventStartDateT
axSchedule1.set_DataField(EXSCHEDULELib.EventKnownPropertyEnum.exEventEndDateTir
axSchedule1.set_DataField(EXSCHEDULELib.EventKnownPropertyEnum.exEventExtraLabel,
axSchedule1.DataSource = (rs as ADODB.Recordset);
```

axSchedule1.EndUpdate();

X++ (Dynamics Ax 2009)

```
// Error event - Fired when an internal error occurs.
void onEvent_Error(int _Error,str _Description)
```

```
;
print(_Description);
}
public void init()
{
anytype rs;
str var_s;
;
super();
```

```
// Add 'Microsoft ActiveX Data Objects 6.0 Library' reference to your project.
rs = COM::createFromObject(new ADODB.Recordset()); rs = rs;
var_s = "Provider=Microsoft.ACE.OLEDB.12.0;Data Source=C:\\Program
Files\\Exontrol\\ExSchedule\\Sample\\Access2007\\datasource.accdb";
```

```
rs.Open("Events",COMVariant::createFromStr(var_s),3/*adOpenStatic*/,3/*adLockOptimistic<sup>2</sup>
```

```
exschedule1.BeginUpdate();
```

```
exschedule1.Calendar().Selection(COMVariant::createFromDate(str2Date("11/11/2013",213)
```

```
exschedule1.DataField(1/*exEventStartDateTime*/,"Start");
exschedule1.DataField(2/*exEventEndDateTime*/,"End");
exschedule1.DataField(11/*exEventExtraLabel*/,"Extra");
exschedule1.DataSource(rs);
exschedule1.EndUpdate();
```

}

VFP

```
*** Error event - Fired when an internal error occurs. ***
LPARAMETERS Error,Description
with thisform.Schedule1
DEBUGOUT( Description )
endwith
```

```
with thisform.Schedule1
rs = CreateObject("ADOR.Recordset")
with rs
.Open("Events","Provider=Microsoft.ACE.OLEDB.12.0;Data Source=C:\Program
Files\Exontrol\ExSchedule\Sample\Access2007\datasource.accdb",3,3)
endwith
.BeginUpdate
.Calendar.Selection = {^2013-11-11}
.Object.DataField(1) = "Start"
.Object.DataField(2) = "End"
.Object.DataField(11) = "Extra"
.DataSource = rs
.EndUpdate
endwith
```

dBASE Plus

```
/*
with (this.ACTIVEX1.nativeObject)
  Error = class::nativeObject_Error
endwith
*/
// Fired when an internal error occurs.
function nativeObject_Error(Error,Description)
  local oSchedule
  oSchedule = form.Activex1.nativeObject
  ? Str(Description)
return
local oSchedule,rs
oSchedule = form.Activex1.nativeObject
rs = new OleAutoClient("ADOR.Recordset")
  rs.Open("Events", "Provider=Microsoft.ACE.OLEDB.12.0;Data Source=C:\Program
Files\Exontrol\ExSchedule\Sample\Access2007\datasource.accdb",3,3)
oSchedule.BeginUpdate()
```

```
oSchedule.Calendar.Selection = "11/11/2013"
oSchedule.Template = [DataField(1) = "Start"] // oSchedule.DataField(1) = "Start"
oSchedule.Template = [DataField(2) = "End"] // oSchedule.DataField(2) = "End"
oSchedule.Template = [DataField(11) = "Extra"] // oSchedule.DataField(11) = "Extra"
oSchedule.EndUpdate()
```

XBasic (Alpha Five)

```
' Fired when an internal error occurs.
function Error as v (Error as N, Description as C)
  Dim oSchedule as P
  oSchedule = topparent:CONTROL_ACTIVEX1.activex
  ? Description
end function
Dim oSchedule as P
Dim rs as P
oSchedule = topparent:CONTROL_ACTIVEX1.activex
rs = OLE.Create("ADOR.Recordset")
  rs.Open("Events", "Provider=Microsoft.ACE.OLEDB.12.0;Data Source=C:\Program
Files\Exontrol\ExSchedule\Sample\Access2007\datasource.accdb",3,3)
oSchedule.BeginUpdate()
oSchedule.Calendar.Selection = {11/11/2013}
oSchedule.Template = "DataField(1) = \"Start\"" ' oSchedule.DataField(1) = "Start"
oSchedule.Template = "DataField(2) = \"End\"" ' oSchedule.DataField(2) = "End"
oSchedule.Template = "DataField(11) = \"Extra\"" ' oSchedule.DataField(11) = "Extra"
oSchedule.DataSource = rs
oSchedule.EndUpdate()
```

Delphi 8 (.NET only)

```
// Error event - Fired when an internal error occurs.
procedure TWinForm1.AxSchedule1_Error(sender: System.Object; e:
AxEXSCHEDULELib._IScheduleEvents_ErrorEvent);
begin
with AxSchedule1 do
```

```
begin
    OutputDebugString(e.description);
  end
end;
with AxSchedule1 do
begin
  rs := (ComObj.CreateComObject(ComObj.ProgIDToClassID('ADOR.Recordset')) as
ADODB.Recordset);
  with rs do
  begin
    Open('Events','Provider=Microsoft.ACE.OLEDB.12.0;Data Source=C:\Program
Files\Exontrol\ExSchedule\Sample\Access2007\datasource.accdb',3,3,Nil);
  end;
  BeginUpdate();
  Calendar.Selection := '11/11/2013';
set_DataField(EXSCHEDULELib.EventKnownPropertyEnum.exEventStartDateTime,'Start');
  set_DataField(EXSCHEDULELib.EventKnownPropertyEnum.exEventEndDateTime,'End');
  set_DataField(EXSCHEDULELib.EventKnownPropertyEnum.exEventExtraLabel,'Extra');
```

DataSource := (rs as ADODB.Recordset);

EndUpdate();

end

Delphi (standard)

```
// Error event - Fired when an internal error occurs.
procedure TForm1.Schedule1Error(ASender: TObject; Error : Integer;Description :
WideString);
begin
with Schedule1 do
begin
OutputDebugString( Description );
end
end;
```

with Schedule1 do

```
begin
  rs :=
(IUnknown(ComObj.CreateComObject(ComObj.ProgIDToClassID('ADOR.Recordset'))) as
ADODB_TLB.Recordset);
  with rs do
  begin
    Open('Events','Provider=Microsoft.ACE.OLEDB.12.0;Data Source=C:\Program
Files\Exontrol\ExSchedule\Sample\Access2007\datasource.accdb',3,3,Null);
  end;
  BeginUpdate();
  Calendar.Selection := '11/11/2013';
  DataField[EXSCHEDULELib TLB.exEventStartDateTime] := 'Start';
  DataField[EXSCHEDULELib TLB.exEventEndDateTime] := 'End';
  DataField[EXSCHEDULELib_TLB.exEventExtraLabel] := 'Extra';
  DataSource := (IUnknown(rs) as ADODB_TLB.Recordset);
  EndUpdate();
end
```

Visual Objects

METHOD OCX_Exontrol1Error(Error,Description) CLASS MainDialog // Error event - Fired when an internal error occurs. OutputDebugString(String2Psz(AsString(Description)))

RETURN NIL

local rs as _Recordset

// Generate Source for 'Microsoft ActiveX Data Objects 6.0 Library' server from Tools\Automation Server...

rs := _Recordset{"ADOR.Recordset"}
rs:Open("Events","Provider=Microsoft.ACE.OLEDB.12.0;Data Source=C:\Program
Files\Exontrol\ExSchedule\Sample\Access2007\datasource.accdb",3,3,0)
oDCOCX_Exontrol1:BeginUpdate()
oDCOCX_Exontrol1:Calendar:Selection := SToD("20131111")
oDCOCX_Exontrol1:[DataField,exEventStartDateTime] := "Start"
oDCOCX_Exontrol1:[DataField,exEventEndDateTime] := "End"
oDCOCX_Exontrol1:[DataField,exEventExtraLabel] := "Extra"
```
oDCOCX_Exontrol1:DataSource := _Recordset{rs}
oDCOCX_Exontrol1:EndUpdate()
```

PowerBuilder

```
/*begin event Error(long Error, string Description) - Fired when an internal error occurs.*/
  OleObject oSchedule
  oSchedule = ole_1.Object
  MessageBox("Information", string(String(Description)))
*/
/*end event Error*/
OleObject oSchedule,rs
oSchedule = ole_1.Object
rs = CREATE OLEObject
rs.ConnectToNewObject("ADOR.Recordset")
  rs.Open("Events", "Provider=Microsoft.ACE.OLEDB.12.0;Data Source=C:\Program
Files\Exontrol\ExSchedule\Sample\Access2007\datasource.accdb",3,3)
oSchedule.BeginUpdate()
oSchedule.Calendar.Selection = 2013-11-11
oSchedule.DataField(1,"Start")
oSchedule.DataField(2,"End")
oSchedule.DataField(11,"Extra")
oSchedule.DataSource = rs
oSchedule.EndUpdate()
```

Visual DataFlex

// Fired when an internal error occurs.

Procedure OnComError Integer IIError String IIDescription Forward Send OnComError IIError IIDescription ShowIn IIDescription End_Procedure

Procedure OnCreate Forward Send OnCreate

Variant rs Get Comcreateobject "ADOR.Recordset" to rs Send ComOpen "Events" "Provider=Microsoft.ACE.OLEDB.12.0;Data Source=C:\Program Files\Exontrol\ExSchedule\Sample\Access2007\datasource.accdb" OLEadOpenStatic OLEadLockOptimistic Nothing Send ComBeginUpdate Variant voCalendar Get ComCalendar to voCalendar Handle hoCalendar Get Create (RefClass(cComCalendar)) to hoCalendar Set pvComObject of hoCalendar to voCalendar Set ComSelection of hoCalendar to "11/11/2013" Send Destroy to hoCalendar Set ComDataField OLEexEventStartDateTime to "Start" Set ComDataField OLEexEventEndDateTime to "End" Set ComDataField OLEexEventExtraLabel to "Extra" Set ComDataSource to rs Send ComEndUpdate End_Procedure

Xbase++

```
PROCEDURE OnError(oSchedule,Error,Description)
    DevOut(Transform(Description,""))
RETURN
#include "AppEvent.ch"
#include "ActiveX.ch"
PROCEDURE Main
LOCAL oForm
LOCAL oForm
LOCAL nEvent := 0, mp1 := NIL, mp2 := NIL, oXbp := NIL
LOCAL oSchedule
LOCAL rs
oForm := XbpDialog():new(AppDesktop())
oForm:drawingArea:clipChildren := .T.
```

```
oForm:create( ,,{100,100}, {640,480},, .F. )
oForm:close := {|| PostAppEvent( xbeP_Quit )}
```

```
oSchedule := XbpActiveXControl():new( oForm:drawingArea )
    oSchedule:CLSID := "Exontrol.Schedule.1" /*{9B09E13D-7A88-4299-9DBE-
383380435377}*/
    oSchedule:create(,, {10,60},{610,370} )
```

oSchedule:Error := {|Error,Description| OnError(oSchedule,Error,Description)} /*Fired when an internal error occurs.*/

```
rs := CreateObject("ADOR.Recordset")
rs:Open("Events","Provider=Microsoft.ACE.OLEDB.12.0;Data Source=C:\Program
Files\Exontrol\ExSchedule\Sample\Access2007\datasource.accdb",3/*adOpenStatic*/,3/*ad
```

```
oSchedule:BeginUpdate()
oSchedule:Calendar():Selection := "11/11/2013"
oSchedule:SetProperty("DataField",1/*exEventStartDateTime*/,"Start")
oSchedule:SetProperty("DataField",2/*exEventEndDateTime*/,"End")
oSchedule:SetProperty("DataField",11/*exEventExtraLabel*/,"Extra")
oSchedule:DataSource := rs
oSchedule:EndUpdate()
```

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

property Schedule.DateEvents (Date as Variant) as Variant

Returns a safe array of Event objects in a giving date.

| Туре | Description |
|-----------------|---|
| Date as Variant | A DATE expression that specifies the date/day to be
queried for events |
| Variant | A Safe array of <u>Event</u> objects being shown in the specified date. |

The DateEvents property returns a collection of events in the specified date. *The /NET and /WPF versions of the component provide the get_DateEvents function that retrieves a collection of Event objects, as List<Event>.* The <u>Start/End</u> properties of the Event object indicates the margins of the events. Once the user starts selecting a new date in the calendar panel, the control fires the <u>LayoutStartChanging(exCalendarSelectionChange)</u>. Once a new date is selected, the <u>LayoutEndChanging(exCalendarSelectionChange)</u> event occurs. The <u>SingleSel</u> property indicates whether the user can select one or multiple dates. The <u>VFP uses the TemplateDef</u> and <u>ExecuteTemplate</u> methods to call the control's properties with parameters, as shown bellow at VFP section.

The following sample shows how you can enumerate the events within the selected date, once the LayoutEndChanging(exCalendarSelectionChange) event occurs.

VB

```
Private Sub Schedule1_LayoutEndChanging(ByVal Operation As
EXSCHEDULELibCt1.LayoutChangingEnum)
If Operation = exCalendarSelectionChange Then
Dim d As Variant
If (Schedule1.Calendar.SelCount = 1) Then
For Each d In Schedule1.DateEvents(Schedule1.Calendar.SelDate(0))
Debug.Print "Event: " & d.Start & " " & d.End
Next
End If
End If
End If
```

VB/NET

Private Sub Exschedule1_LayoutEndChanging(ByVal sender As System.Object, ByVal

```
Operation As exontrol.EXSCHEDULELib.LayoutChangingEnum) Handles
Exschedule1.LayoutEndChanging
If Operation =
exontrol.EXSCHEDULELib.LayoutChangingEnum.exCalendarSelectionChange Then
If (Exschedule1.Calendar.SelCount > 0) Then
Dim evs As List(Of exontrol.EXSCHEDULELib.Event) =
Exschedule1.get_DateEvents(Exschedule1.Calendar.get_SelDate(0))
If Not evs Is Nothing Then
For Each d As exontrol.EXSCHEDULELib.Event In evs
Debug.Print("Event: " & d.Start & " " & d.End)
Next
End If
End If
End If
End Sub
```

or:

| Private Sub Exschedule1_LayoutEndChanging(ByVal sender As System.Object, ByVal |
|--|
| Operation As exontrol.EXSCHEDULELIb.LayoutChangingEnum) Handles |
| Exschedule1.LayoutEndChanging |
| If Operation = |
| exontrol.EXSCHEDULELib.LayoutChangingEnum.exCalendarSelectionChange Then |
| If (Exschedule1.Calendar.SelCount > 0) Then |
| Dim evs As List(Of exontrol.EXSCHEDULELib.Event) = |
| Exschedule1.get_DateEvents(Exschedule1.Calendar.get_SelDate(0)) |
| If Not evs Is Nothing Then |
| For Each d As exontrol.EXSCHEDULELib.Event In evs |
| Debug.Print("Event: " & d.Start & " " & d.End) |
| Next |
| End If |
| End If |
| End If |
| End Sub |
| |

C#

private void exschedule1_LayoutEndChanging(object sender,

```
exontrol.EXSCHEDULELib.LayoutChangingEnum Operation)
{
    if ( Operation ==
    exontrol.EXSCHEDULELib.LayoutChangingEnum.exCalendarSelectionChange )
        if ( exschedule1.Calendar.SelCount == 1 )
        {
            List<exontrol.EXSCHEDULELib.Event> evs =
        exschedule1.get_DateEvents(exschedule1.Calendar.get_SelDate(0));
            if ( evs != null )
            foreach (exontrol.EXSCHEDULELib.Event d in evs)
            System.Diagnostics.Debug.Print("Event: " + d.Start.ToString() + " " +
        d.Start.ToString());
        }
    }
}
```

```
VFP
```

```
*** ActiveX Control Event ***
LPARAMETERS operation
* 1 ' exCalendarSelectionChange
  If Operation = 1 Then
    IF (thisform.Schedule1.Calendar.SelCount = 1) Then
      local e, evs as Object
      thisform.Schedule1.TemplateDef = "dim d"
      thisform.Schedule1.TemplateDef = thisform.Schedule1.Calendar.SelDate(0)
      evs = thisform.Schedule1.ExecuteTemplate("DateEvents(d)")
      For Each e In evs
        LOCAL ee as Object
        ee = thisform.Schedule1.Events(e)
        WAIT WINDOW TTOC(ee.Start) + " " + TTOC(ee.End)
      ENDFOR
    ENDif
  Endlf
```

C++

void LayoutEndChangingSchedule1(long Operation)

```
if (Operation = = EXSCHEDULELib::exCalendarSelectionChange)
    if (m_spSchedule->Calendar->SelCount = = 1)
    {
      _variant_t evs;
      if (SUCCEEDED(m_spSchedule->get_DateEvents(_variant_t(m_spSchedule-
>Calendar->SelDate[0], VT_DATE ), &evs ) ) )
        if (V_VT( &evs ) = = ( VT_ARRAY | VT_VARIANT ) )
        ł
          BYTE* p = NULL;
          long nCount = 0;
          if (SUCCEEDED(SafeArrayGetUBound(V_ARRAY(&evs), 1, &nCount)))
          {
             if ( SUCCEEDED( SafeArrayAccessData( V_ARRAY( &evs ), (LPVOID*)&p ) ) )
             ł
               for (long i = 0; i < nCount + 1; i++, p += sizeof(VARIANT))
               {
                 VARIANT* pValue = (VARIANT*)p;
                 if (V_VT(pValue) = VT_DISPATCH)
                 {
                   EXSCHEDULELib::IEventPtr spEvent = V_DISPATCH( pValue );
                   CString strMessage;
                   strMessage.Format(_T("Event: %f %f\r\n"), spEvent->Start, spEvent-
>End );
                   OutputDebugString( strMessage );
                 }
               SafeArrayUnaccessData( V_ARRAY( &evs ) );
            }
          }
        }
    }
```

where m_spSchedule is of EXSCHEDULELib::ISchedulePtr type.

property Schedule.DateTimeFromPoint (X as OLE_XPOS_PIXELS, Y as OLE_YPOS_PIXELS) as Date

Retrieves the date/time from the cursor, in the schedule panel.

| Туре | Description |
|----------------------|--|
| X as OLE_XPOS_PIXELS | A single that specifies the current X location of the mouse
pointer. The x values is always expressed in client
coordinates. |
| Y as OLE_YPOS_PIXELS | A single that specifies the current Y location of the mouse
pointer. The y values is always expressed in client
coordinates. |
| Date | A Date expression that indicates the date from the cursor. |

The MouseMove event is generated continually as the mouse pointer moves across objects. During the MouseMove event you can call the ShowToolTip method to display any custom tooltip. During the Click or RClick event you can get an UI part of the control using one of the following properties. All ...FromPoint properties can be use such as ...FromPoint(-1,-1) to get the UI part of the control from the current mouse position, in other words, you do not have to pass any X, Y coordinates. The DateTimeFromPoint property gets the date including the time. You can use any function that gets the integer part of the date, to get the date only, not including the time, such as Fix, DateValue, ... in VB. For instance, the VB has a Fix method that gets the integer part of the number. So, the Fix(DateTimeFromPoint(-1,-1)) gets the date from the cursor, not including the time as shown in the time scale.

You can get UI parts from the cursor, using any of the following ... FromPoint properties:

- Calendar. <u>DateFromPoint</u> property gets the date from the cursor in the Calendar panel.
- <u>AnchorFromPoint</u> property gets the anchor (<a>) element from the cursor.
- <u>DateTimeFromPoint</u> property gets the date/time from the cursor, in the Schedule panel.
- <u>TimeFromPoint</u> property gets the time from the cursor, in the Schedule panel.
- EventFromPoint property gets the Event object from the cursor.
- <u>GroupFromPoint</u> property retrieves the <u>Group</u> object from the cursor. Use the GroupHeaderFromPoint property to the Group object when cursor hovers the group's header.
- <u>GroupHeaderFromPoint</u> property retrieves the <u>Group</u> object from the cursor, when the cursor hovers just the group's header.
- <u>MarkTimeFromPoint</u> property gets the <u>MarkTime</u> object from the cursor. A MarkTime object is identified by a date/time where it is displaying.
- <u>MarkZoneFromPoint</u> property gets the <u>MarkZone</u> object from the cursor. A MarkZone object is identified by a starting date/time and ending date/time where it is displaying.

- <u>NonworkingTimeFromPoint</u> property indicates the <u>NonworkingTime</u> object from the cursor.
- <u>PictureFromPoint</u> property indicates the key of the picture from the cursor.
- <u>TimeScaleFromPoint</u> property returns the <u>TimeScale</u> object from the cursor.

All ... FromPoint properties that returns an Object, may return Nothing, Empty or NULL, if no object is found.

property Schedule.DayEndTime as String

Indicates the day end time.

| Туре | Description |
|--------|--|
| String | A String expression that indicates the ending time of the day. For instance, the "08:00" indicates 08:00 AM, while the "16:00" indicates the 08:00 PM. |

By default, the DayEndTime property is "16:00", which indicates that the day ends at 8:00 PM. The <u>DayStartTime</u> property specifies the time to start the day. The <u>TimeZone</u> property can be used to programmatically update the time zone. The <u>MajorTimeRuler</u> property indicates the time to increment the major rulers, while the <u>MinorTimeRuler</u> property specifies the time to increment the minor rulers. The <u>DateTimeFromPoint/TimeFromPoint</u> properties can be used to get the date/time from the cursor.

property Schedule.DayStartTime as String

Indicates the day start time.

| Туре | Description |
|--------|--|
| String | A String expression that indicates the starting time of the day. For instance, the "08:00" indicates 08:00 AM, while the "16:00" indicates the 08:00 PM. |

By default, the DayStartTime property is "08:00", which indicates that the day starts at 8:00 AM. The <u>DayEndTime</u> property specifies the time to end the day. The <u>TimeZone</u> property can be used to programmatically update the time zone. The <u>MajorTimeRuler</u> property indicates the time to increment the major rulers, while the <u>MinorTimeRuler</u> property specifies the time to increment the minor rulers. The <u>DateTimeFromPoint/TimeFromPoint</u> properties can be used to get the date/time from the cursor.

property Schedule.DayViewHeight as Long

l

Indicates the height of the day's view in the schedule panel.

| Туре | Description |
|------|---|
| Long | A Long expression that specifies the height of the date in the schedule view. |

The DayViewHeight property specifies the height, in pixels, of the date in the schedule panel. If setting the DayViewHeight property on -1 (negative value), the date fits vertically the schedule view.

The <u>DayViewOffsetX</u> property indicates the horizontal scroll position of the schedule's view. The <u>DayViewWidth</u> property specifies the width, in pixels, of the date in the schedule panel. The <u>DayViewOffsetY</u> property indicates the vertical scroll position of the schedule's view. You can use the DayViewWidth, <u>DayViewHeight</u>, <u>DayViewOffsetX</u> and <u>DayViewOffsetY</u> properties to save and restore the schedule view position. The <u>EnsureVisible</u> method ensures that giving date fits the schedule's view.

property Schedule.DayViewOffsetX as Long

Indicates the horizontal scroll position of the schedule's view.

| Туре | Description |
|------|---|
| Long | A Long expression that specifies the horizontal scroll
position of the schedule's view |

The DayViewOffsetX property indicates the horizontal scroll position of the schedule's view. If setting the DayViewOffsetX property on -1 (negative value), the schedule view scrolls to the end (horizontally).

The <u>DayViewWidth</u> property specifies the width, in pixels, of the date in the schedule panel. The <u>DayViewHeight</u> property specifies the height, in pixels, of the date in the schedule panel. The <u>DayViewOffsetY</u> property indicates the vertical scroll position of the schedule's view. You can use the <u>DayViewWidth</u>, <u>DayViewHeight</u>, DayViewOffsetX and <u>DayViewOffsetY</u> properties to save and restore the schedule view position. The <u>EnsureVisible</u> method ensures that giving date fits the schedule's view.

property Schedule.DayViewOffsetY as Long

Indicates the vertical scroll position of the schedule's view.

| Туре | Description |
|------|--|
| Long | A Long expression that specifies the vertical scroll position of the schedule's view |

The DayViewOffsetY property indicates the vertical scroll position of the schedule's view. If setting the DayViewOffsetY property on -1 (negative value), the schedule view scrolls to the end (vertically).

The <u>DayViewWidth</u> property specifies the width, in pixels, of the date in the schedule panel. The <u>DayViewHeight</u> property specifies the height, in pixels, of the date in the schedule panel. The <u>DayViewOffsetX</u> property indicates the horizontal scroll position of the schedule's view. You can use the <u>DayViewWidth</u>, <u>DayViewHeight</u>, DayViewOffsetX and <u>DayViewOffsetY</u> properties to save and restore the schedule view position. The <u>EnsureVisible</u> method ensures that giving date fits the schedule's view.

property Schedule.DayViewWidth as Long

Indicates the width of the day's view in the schedule panel.

| Туре | Description |
|------|--|
| Long | A Long expression that specifies the width of the date in the schedule view. |

The DayViewWidth property specifies the width, in pixels, of the date in the schedule panel. If setting the DayViewWidth property on -1 (negative value), the date fits horizontally the schedule view.

The <u>DayViewOffsetX</u> property indicates the horizontal scroll position of the schedule's view. The <u>DayViewHeight</u> property specifies the height, in pixels, of the date in the schedule panel. The <u>DayViewOffsetY</u> property indicates the vertical scroll position of the schedule's view. You can use the DayViewWidth, <u>DayViewHeight</u>, <u>DayViewOffsetX</u> and <u>DayViewOffsetY</u> properties to save and restore the schedule view position. The <u>EnsureVisible</u> method ensures that giving date fits the schedule's view.

property Schedule.DefaultEventLongLabel as String

Indicates the default long label for events.

| Туре | Description |
|--------|--|
| String | A string expression that specifies the extended HTML label, to be displayed on the event's body. |

By default, the DefaultEventLongLabel property is "<%=%256%>", which indicates that the events displays its margins in a short format. The DefaultEventLongLabel property indicates the default long label for events. The LongLabel property is displayed only if it fits the event's body, else the <u>ShortLabel</u> property is shown. For instance, the ShortLabel property is shown if the event's body is too small. The event displays the <u>ExtraLabel</u>, only if the LongLabel property is displayed. The <u>AddEvent</u> event occurs once a new event is added to the Events collection.

The <u>DefaultEventPadding</u> property indicates the padding of the labels on the event, relative to event's borders. The <u>DefaultEventShortLabel</u> property defines the initial value for the event's <u>ShortLabel</u> property. The <u>DefaultEventTooltip</u> property defines the event's tooltip.

Here's a few samples:

l

- "new", simple new text is shown.
- "<a *no*>title", displays a clickable text such as <u>title</u>, and <u>AnchorClick</u> can be used to determine whether the *no* anchor has been clicked.
- "<a>pic1:32", displays a click able image, the <u>AnchorClick</u> can be used to determine whether the anchor has been clicked. We would recommend using the <u>Pictures</u> or <u>ExtraPictures</u> property to assign pictures to an event.
- "<%=%256%>", displays the event's start and end points in a short format.
- "<%=%257%>", displays the event's margins in a long format.
- "Start: <%=%1%>
End: <%=%2%>", displays the starting margin of the even on the first line, while on the second line it displays the ending point of the event.
- "<%=%256%>
Caption: <%=%5%>", displays the event's margins in short format on the first line, and on the second line it displays the event's <u>Caption</u> property. The caption shown on the event's body is automatically updated once the event is moved to a new position or the event's Caption is changed.
- "<%=%256%>
<%=%264? `repetitive event`:``%>" displays automatically the "repetitive event" for repetitive events, or when the event's <u>Repetitive</u> property is not empty and valid
- "Duration: <%=((1:=int(0:= (date(%2)-date(%1)))) != 0 ? (=:1 + ' day(s)') : ") + (=:1 ? ' ' : ") + ((1:=int(0:=((=:0 =:1 + 1/24/60/60/2)*24))) != 0 ? =:1 + ' hour(s)' : ") + (=:1 ? ' : ") + ((1:=round((=:0 =:1)*60)) != 0 ? =:1 + ' min(s)' : ")%>" displays the duration of the event in days, hours and minutes.
- "<%=%256%>
Duration: <%=((1:=int(0:= (date(%2)-date(%1)))) != 0 ? (=:1 + '</p>

 $\begin{aligned} day(s)') : ") + (=:1 ? ' : ") + ((1:=int(0:=((=:0 - =:1 + 1/24/60/60/2)*24)))) != 0 ? =:1 + 1/24/60/60/2)*24))) != 0 ? =:1 + 1/24/60/60/2)*24))) != 0 ? =:1 + 1/24/60/60/2)*24)) != 0 ? =:1 + 1/24/60/60/2) != 0 ? =:1 + 1/24/60/60/2) != 0 ? =:1 + 1/24/60/60/2) != 0 ? =:1 + 1/24/60/60/2) != 0 ? =:1 + 1/24/60/60/2) != 0 ? =:1 + 1/24/60/60/2) != 0 ? =:1 + 1/24/60/60/2) != 0 ? =:1 + 1/24/60/60/2) != 0 ? =:1 + 1/24/60/60/2) != 0 != 0 ! != 0 != 0 != 0 != 0 ! != 0 !=$

")%>" displays the event's margins on the first line and the duration of the event in days, hours and minutes, on the second line

"<%=%><%=%5%>
<%=%256%>", displays the event's Caption on first line(s), following by the event's Start/End margins in short date-time format. The <%=%> prefix forces the expression to be re-evaluated and apply any HTML tag found. For instance, %5 indicates the event's Caption property, and if it contains HTML tags they will be applied as is, instead displaying them as a plain text. Any expression that starts with "<%=%>" is re- evaluated and its result is displayed in HTML format (available starting with the version 12.2)

The <u>EventKnowPropertyEnum</u> defines the %identifiers that can be used in formula <%=FORMULA%>. For instance, the Label property on "Start: <%=time(%1) replace `AM` with ``%>" displays the time when the event starts with no AM time indicators.

The property supports the following identifiers. These identifiers can be used in FORMULA format:

- **%1**, Indicates the starting date/time of the event as DATE type, equivalent with <u>Start</u> property
- %2, Indicates the ending date/time of the event as DATE type, equivalent with <u>End</u> property
- %3, Indicates if the current event is an all day event as BOOL type, equivalent with <u>AllDayEvent</u> property
- %4, Indicates the identifier of the event's group, as LONG type, equivalent with <u>GroupID</u> property.
- **%5**, Indicates the caption of the event, as STRING expression, equivalent with <u>Caption</u> property.
- %6, Indicates the extra data associated with the event, as VARIANT type, equivalent with <u>UserData</u> property.
- %7, Gets or sets the duration of the event as FLOAT expression. Above you can find how you can display the duration of the event in hours, minutes...
- %8, Specifies the repetitive expression of the event, equivalent with <u>Repetitive</u> property.
- %256, Gets the margins of the event in a short format, as a STRING expression. The <u>ShortDateFormat</u> property defines the short date format. The <u>ShortTimeFormat</u> property defines the short time format.
- %257, Gets the margins of the event in a long format, as a STRING expression. The <u>LongDateFormat</u> property defines the long date format. The <u>LongTimeFormat</u> property defines the long time format.
- %258, Gets the starting date (not including the time) of the current event, as a DATE

type.

- %259, Gets the starting time (not including the date) of the current event, as DATE type from 0 to 1.
- **%260**, Gets the ending date (not including the time) of the current event, as a DATE type.
- **%261**, Gets the ending time (not including the date) of the current event, as DATE type from 0 to 1.
- **%262**, Gets the label of the owner group, as STRING expression, equivalent with <u>Caption</u> property of the Group's event.
- %263, Gets the title of the owner group, as STRING expression, equivalent with <u>Title</u> property of the Group's event.
- %264, Indicates if the current event is a repetitive event, as BOOL type. This flag returns TRUE, if the <u>Repetitive</u> property is not empty, and the expression is valid.

The FORMULA, is identified by <%=FORMULA%>, and supports the following predefined operators and functions:

The supported binary arithmetic operators are:

- * (multiplicity operator), priority 5
- *I* (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#, #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
- Itrim (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 property supports the following built-in HTML tags:

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

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

- exp, stores the plain text to be shown once the user clicks the anchor, such as "<a ;exp=show lines>"
- e64, encodes in BASE64 the HTML text to be shown once the user clicks the anchor, such as "<a ;e64=gA8ABmABnABjABvABshIAOQAEAAHAAGESikWio+ABzABohp3iELABpABu " that displays show lines- in gray when the user clicks the + anchor. The "gA8ABmABnABjABvABshIAOQAEAAHAAGESikWio+ABzABohp3iELABpABuABljY

string encodes the "<fgcolor 808080>show lines<a>-</fgcolor>" The Decode64Text/Encode64Text methods of the eXPrint can be used to decode/encode e64 fields.

Any ex-HTML caption can be transformed to an expandable-caption, by inserting the anchor ex-HTML tag. For instance, "<solidline>Header</solidline>

Line1<r><a ;exp=show lines>+
Line2
Line3" shows the Header in underlined and bold on the first line and Line1, Line2, Line3 on the rest. The "show lines" is shown instead of Line1, Line2, Line3 once the user clicks the + sign.

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

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

Returns or sets a value that indicates the padding of the events in the control.

| Туре | Description |
|--------------------------------|--|
| Edge as <u>PaddingEdgeEnum</u> | A PaddingEdgeEnum expression that indicates the edge to be modified. |
| Long | A long expression that defines the padding. |

The DefaultEventPadding property indicates the padding of the labels on the event, relative to event's borders. The <u>DefaultEventLongLabel</u> property indicates the default long label for events. The <u>DefaultEventShortLabel</u> property defines the initial value for the event's <u>ShortLabel</u> property. The <u>DefaultEventTooltip</u> property defines the event's tooltip. The <u>LongLabel</u> property is displayed only if it fits the event's body, else the <u>ShortLabel</u> property is shown. For instance, the ShortLabel property is shown if the event's body is too small. The event displays the <u>ExtraLabel</u>, only if the LongLabel property is displayed. The <u>AddEvent</u> event occurs once a new event is added to the Events collection.

property Schedule.DefaultEventShortLabel as String

Indicates the default short label for events.

| Туре | Description |
|--------|--|
| String | A string expression that specifies the extended HTML label, to be displayed on the event's body. <i>The images, or any font HTML attribute is ignored.</i> |

By default, the DefaultEventShortLabel property is "<%=%256%>", which means that the event's body displays the margins of events in short format . The DefaultEventShortLabel property defines the initial value for event's <u>ShortLabel</u> property. The <u>ShortLabel</u> property is shown if the event's body is too small. The <u>LongLabel</u> property is displayed only if it fits the event's body, else the ShortLabel property is shown. The event displays the <u>ExtraLabel</u>, only if the LongLabel property is displayed. The <u>AddEvent</u> event occurs once a new event is added to the Events collection. The <u>LabelAlign</u> property specifies the alignment of the long label. *The DefaultEventShortLabel/ShortLabel property displays (ignores) NO images such as , or font HTML attributes such as , <i>, ...*

The <u>DefaultEventPadding</u> property indicates the padding of the labels on the event, relative to event's borders. The <u>DefaultEventLongLabel</u> property defines the initial value for the event's <u>LongLabel</u> property. The <u>DefaultEventTooltip</u> property defines the event's tooltip.

Here's a few samples:

- "new", simple new text is shown.
- "<%=%256%>", displays the event's start and end points in a short format.
- "<%=%257%>", displays the event's margins in a long format.
- "Start: <%=%1%>
End: <%=%2%>", displays the starting margin of the even on the first line, while on the second line it displays the ending point of the event.
- "<%=%256%>
Caption: <%=%5%>", displays the event's margins in short format on the first line, and on the second line it displays the event's <u>Caption</u> property. The caption shown on the event's body is automatically updated once the event is moved to a new position or the event's Caption is changed.
- "<%=%256%>
<%=%264? `repetitive event`:``%>" displays automatically the "repetitive event" for repetitive events, or when the event's <u>Repetitive</u> property is not empty and valid
- "Duration: <%=((1:=int(0:= (date(%2)-date(%1)))) != 0 ? (=:1 + ' day(s)') : ") + (=:1 ? ' ' : ") + ((1:=int(0:=((=:0 =:1 + 1/24/60/60/2)*24))) != 0 ? =:1 + ' hour(s)' : ") + (=:1 ? ' ' : ") + ((1:=round((=:0 =:1)*60)) != 0 ? =:1 + ' min(s)' : ")%>" displays the duration of the event in days, hours and minutes.

' hour(s)': ") + (=:1?'': ") + ((1:=round((=:0 - =:1)*60)) != 0? =:1 + ' min(s)': ")%>" displays the event's margins on the first line and the duration of the event in days, hours and minutes, on the second line

"<%=%><%=%5%>
<%=%256%>", displays the event's Caption on first line(s), following by the event's Start/End margins in short date-time format. The <%=%> prefix forces the expression to be re-evaluated and apply any HTML tag found. For instance, %5 indicates the event's Caption property, and if it contains HTML tags they will be applied as is, instead displaying them as a plain text. Any expression that starts with "<%=%>" is re- evaluated and its result is displayed in HTML format (available starting with the version 12.2)

The <u>EventKnowPropertyEnum</u> defines the %identifiers that can be used in formula <%=FORMULA%>. For instance, the ShortLabel property on "Start: <%=time(%1) replace `AM` with ``%>" displays the time when the event starts with no AM time indicators.

The property supports the following identifiers. These identifiers can be used in FORMULA format:

- **%1**, Indicates the starting date/time of the event as DATE type, equivalent with <u>Start</u> property
- %2, Indicates the ending date/time of the event as DATE type, equivalent with <u>End</u> property
- %3, Indicates if the current event is an all day event as BOOL type, equivalent with <u>AllDayEvent</u> property
- %4, Indicates the identifier of the event's group, as LONG type, equivalent with <u>GroupID</u> property.
- **%5**, Indicates the caption of the event, as STRING expression, equivalent with <u>Caption</u> property.
- %6, Indicates the extra data associated with the event, as VARIANT type, equivalent with <u>UserData</u> property.
- %7, Gets or sets the duration of the event as FLOAT expression. Above you can find how you can display the duration of the event in hours, minutes...
- %8, Specifies the repetitive expression of the event, equivalent with <u>Repetitive</u> property.
- %256, Gets the margins of the event in a short format, as a STRING expression. The <u>ShortDateFormat</u> property defines the short date format. The <u>ShortTimeFormat</u> property defines the short time format.
- %257, Gets the margins of the event in a long format, as a STRING expression. The <u>LongDateFormat</u> property defines the long date format. The <u>LongTimeFormat</u> property defines the long time format.
- %258, Gets the starting date (not including the time) of the current event, as a DATE type.
- %259, Gets the starting time (not including the date) of the current event, as DATE

type from 0 to 1.

- **%260**, Gets the ending date (not including the time) of the current event, as a DATE type.
- **%261**, Gets the ending time (not including the date) of the current event, as DATE type from 0 to 1.
- %262, Gets the label of the owner group, as STRING expression, equivalent with <u>Caption</u> property of the Group's event.
- %263, Gets the title of the owner group, as STRING expression, equivalent with <u>Title</u> property of the Group's event.
- %264, Indicates if the current event is a repetitive event, as BOOL type. This flag returns TRUE, if the <u>Repetitive</u> property is not empty, and the expression is valid.

The FORMULA, is identified by <%=FORMULA%>, and supports 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: "<i>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:

- \circ 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
- Itrim (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)
property Schedule.DefaultEventTooltip as String

Indicates the default tooltip for events.

| Туре | Description |
|--------|--|
| String | A String expression that defines the extended HTML format to be displayed when the cursor hovers the appointments. |

By default, the DefaultEventTooltip property is "*Start:* <%=%1%><*br>End:* <%=%2%> <*br>Duration:* <%=((1:=int(0:= (date(%2)-date(%1)))) != 0 ? (=:1 + ' day(s)') : ") + (=:1 ? ' ' : ") + ((1:=int(0:=((=:0 - =:1 + 1/24/60/60/2)*24))) != 0 ? =:1 + ' hour(s)' : ") + (=:1 ? ' ' : ") + ((1:=round((=:0 - =:1)*60)) != 0 ? =:1 + ' min(s)' : ")%>", which indicates that the tooltip of the event shows the start, end and the duration of the event. The DefaultEventTooltip property defines the event's tooltip, or the initial value for the event's <u>ToolTip</u>. The <u>ShowToolTip</u> method can be used during the <u>MouseMove</u> event to display a custom tooltip.

The <u>DefaultEventPadding</u> property indicates the padding of the labels on the event, relative to event's borders. The <u>DefaultEventShortLabel</u> property defines the initial value for the event's <u>ShortLabel</u> property. The <u>DefaultEventLongLabel</u> property defines the initial value for the event's <u>LongLabel</u> property. Use the <u>ToolTipPopDelay</u> property specifies the period in ms of time the ToolTip remains visible if the mouse pointer is stationary within a control. Use the <u>ToolTipFont</u> property to change the tooltip's font.

Here's a few samples:

- "new", simple new text is shown.
- "<%=%256%>", displays the event's start and end points in a short format.
- "<%=%257%>", displays the event's margins in a long format.
- "Start: <%=%1%>
End: <%=%2%>", displays the starting margin of the even on the first line, while on the second line it displays the ending point of the event.
- "<%=%256%>
Caption: <%=%5%>", displays the event's margins in short format on the first line, and on the second line it displays the event's <u>Caption</u> property. The caption shown on the event's body is automatically updated once the event is moved to a new position or the event's Caption is changed.
- "<%=%256%>
<%=%264? `repetitive event`:``%>" displays automatically the "repetitive event" for repetitive events, or when the event's <u>Repetitive</u> property is not empty and valid
- "Duration: <%=((1:=int(0:= (date(%2)-date(%1)))) != 0 ? (=:1 + ' day(s)') : ") + (=:1 ? ' ' : ") + ((1:=int(0:=((=:0 =:1 + 1/24/60/60/2)*24))) != 0 ? =:1 + ' hour(s)' : ") + (=:1 ? ' ' : ") + ((1:=round((=:0 =:1)*60)) != 0 ? =:1 + ' min(s)' : ")%>" displays the duration of the event in days, hours and minutes.

"<%=%256%>
Duration: <%=((1:=int(0:= (date(%2)-date(%1)))) != 0 ? (=:1 + ' day(s)') : ") + (=:1 ? ' ' : ") + ((1:=int(0:=((=:0 - =:1 + 1/24/60/60/2)*24))) != 0 ? =:1 + ' hour(s)' : ") + (=:1 ? ' ' : ") + ((1:=round((=:0 - =:1)*60)) != 0 ? =:1 + ' min(s)' : ")%>" displays the event's margins on the first line and the duration of the event in days, hours and minutes, on the second line

The <u>EventKnowPropertyEnum</u> defines the %identifiers that can be used in formula <%=FORMULA%>. For instance, the ToolTip property on "Start: <%=time(%1) replace `AM` with ``%>" displays the time when the event starts with no AM time indicators.

The property supports the following identifiers. These identifiers can be used in FORMULA format:

- %1, Indicates the starting date/time of the event as DATE type, equivalent with <u>Start</u> property
- %2, Indicates the ending date/time of the event as DATE type, equivalent with End property
- %3, Indicates if the current event is an all day event as BOOL type, equivalent with <u>AllDayEvent</u> property
- %4, Indicates the identifier of the event's group, as LONG type, equivalent with <u>GroupID</u> property.
- **%5**, Indicates the caption of the event, as STRING expression, equivalent with <u>Caption</u> property.
- %6, Indicates the extra data associated with the event, as VARIANT type, equivalent with <u>UserData</u> property.
- %7, Gets or sets the duration of the event as FLOAT expression. Above you can find how you can display the duration of the event in hours, minutes...
- %8, Specifies the repetitive expression of the event, equivalent with <u>Repetitive</u> property.
- %256, Gets the margins of the event in a short format, as a STRING expression. The <u>ShortDateFormat</u> property defines the short date format. The <u>ShortTimeFormat</u> property defines the short time format.
- %257, Gets the margins of the event in a long format, as a STRING expression. The <u>LongDateFormat</u> property defines the long date format. The <u>LongTimeFormat</u> property defines the long time format.
- %258, Gets the starting date (not including the time) of the current event, as a DATE type.
- %259, Gets the starting time (not including the date) of the current event, as DATE type from 0 to 1.
- %260, Gets the ending date (not including the time) of the current event, as a DATE type.
- %261, Gets the ending time (not including the date) of the current event, as DATE

type from 0 to 1.

- %262, Gets the label of the owner group, as STRING expression, equivalent with <u>Caption</u> property of the Group's event.
- %263, Gets the title of the owner group, as STRING expression, equivalent with <u>Title</u> property of the Group's event.
- %264, Indicates if the current event is a repetitive event, as BOOL type. This flag returns TRUE, if the <u>Repetitive</u> property is not empty, and the expression is valid.

The FORMULA, is identified by <%=FORMULA%>, and supports 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#, #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
- Itrim (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 property supports the following built-in HTML tags:

- ... displays the text in **bold**
- <i> ... </i> displays the text in *italics*

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

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

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

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

- ... displays portions of text with a different font and/or different size. For instance, the "bit" draws the bit text using the Tahoma font, on size 12 pt. If the name of the font is missing, and instead size is present, the current font is used with a different size. For instance, "bit" displays the bit text using the current font, but with a different size.
- **<fgcolor rrggbb>** ... **</fgcolor>** or **<**fgcolor=rrggbb> ... **</**fgcolor> displays text with a specified foreground color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- <bgcolor rrggbb> ... </bgcolor> or <bgcolor=rrggbb> ... </bgcolor> displays text with a specified background color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- <solidline rrggbb> ... </solidline> or <solidline=rrggbb> ... </solidline> draws a solid-

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

- <dotline rrggbb> ... </dotline> or <dotline=rrggbb> ... </dotline> draws a dot-line on the bottom side of the current text-line, of specified RGB color. The <dotline> ...
 </dotline> draws a black dot-line on the bottom side of the current text-line. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<upline>** ... **</upline>** draws the line on the top side of the current text-line (requires <solidline> or <dotline>).
- <r> right aligns the text
- <c> centers the text
- **
>** forces a line-break
- number[:width] 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.
- key[:width] 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 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** FFFFF;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 Schedule.Description(Type as DescriptionTypeEnum) as String

Changes descriptions for control objects.

| Туре | Description |
|---------------------------------------|--|
| Type as
<u>DescriptionTypeEnum</u> | A DescriptionTypeEnum expression that defines the UI part to change its caption. |
| String | A string expression that defines the new caption for the giving UI part. |

The Description property defines the default caption to be displayed on giving UI part of the control. For instance, you can use the Description property to change the "All" caption being displayed on the groups filter.

The following samples translate "All" in the drop down filter of the groups to "Todos":

VBA (MS Access, Excell...)

```
With Schedule1

.Description(0) = "(todos)"

.DisplayGroupingButton = True

.ShowGroupingEvents = True

.Groups.Add 1,"Pro"

End With
```

VB6

```
With Schedule1

.Description(exGroupBarAll) = "(todos)"

.DisplayGroupingButton = True

.ShowGroupingEvents = True

.Groups.Add 1,"Pro"

End With
```

VB.NET

With Exschedule1

.set_Description(exontrol.EXSCHEDULELib.DescriptionTypeEnum.exGroupBarAll," (todos)")

.DisplayGroupingButton = True

```
.ShowGroupingEvents = True
```

```
.Groups.Add(1,"Pro")
End With
```

VB.NET for /COM

```
With AxSchedule1

.set_Description(EXSCHEDULELib.DescriptionTypeEnum.exGroupBarAll,"(todos)")

.DisplayGroupingButton = True

.ShowGroupingEvents = True

.Groups.Add(1,"Pro")

End With
```

C++

```
Copy and paste the following directives to your header file as
it defines the namespace 'EXSCHEDULELib' for the library: 'ExSchedule 1.0 Control
Library'
```

```
#import <ExSchedule.dll>
using namespace EXSCHEDULELib;
*/
EXSCHEDULELib::ISchedulePtr spSchedule1 = GetDlgItem(IDC_SCHEDULE1)-
>GetControlUnknown();
spSchedule1->PutDescription(EXSCHEDULE1ib::oxGroupBarAllL!"(todos)");
```

```
spSchedule1->PutDescription(EXSCHEDULELib::exGroupBarAll,L"(todos)");
```

```
spSchedule1->PutDisplayGroupingButton(VARIANT_TRUE);
```

```
spSchedule1->PutShowGroupingEvents(VARIANT_TRUE);
```

```
spSchedule1->GetGroups()->Add(1,L"Pro");
```

C++ Builder

```
Schedule1->Description[Exschedulelib_tlb::DescriptionTypeEnum::exGroupBarAll] =
L"(todos)";
Schedule1->DisplayGroupingButton = true;
Schedule1->ShowGroupingEvents = true;
Schedule1->Groups->Add(1,L"Pro");
```

exschedule1.**set_Description**(exontrol.EXSCHEDULELib.DescriptionTypeEnum.exGroup (todos)"); exschedule1.DisplayGroupingButton = true; exschedule1.ShowGroupingEvents = true; exschedule1.Groups.Add(1,"Pro");

JavaScript

```
<OBJECT classid="clsid:9B09E13D-7A88-4299-9DBE-383380435377"
id="Schedule1"></OBJECT>
```

<SCRIPT LANGUAGE="JScript"> Schedule1.**Description**(0) = "(todos)"; Schedule1.DisplayGroupingButton = true; Schedule1.ShowGroupingEvents = true; Schedule1.Groups.Add(1,"Pro"); </SCRIPT>

C# for /COM

```
axSchedule1.set_Description(EXSCHEDULELib.DescriptionTypeEnum.exGroupBarAll,"
(todos)");
axSchedule1.DisplayGroupingButton = true;
axSchedule1.ShowGroupingEvents = true;
axSchedule1.Groups.Add(1,"Pro");
```

X++ (Dynamics Ax 2009)

```
public void init()
{
  ;
  super();
  exschedule1.Description(0/*exGroupBarAll*/,"(todos)");
```

C#

```
exschedule1.DisplayGroupingButton(true);
exschedule1.ShowGroupingEvents(true);
exschedule1.Groups().Add(1,"Pro");
```

Delphi 8 (.NET only)

```
with AxSchedule1 do
begin
set_Description(EXSCHEDULELib.DescriptionTypeEnum.exGroupBarAll,'(todos)');
DisplayGroupingButton := True;
ShowGroupingEvents := True;
Groups.Add(1,'Pro');
end
```

Delphi (standard)

```
with Schedule1 do
begin
    Description[EXSCHEDULELib_TLB.exGroupBarAll] := '(todos)';
    DisplayGroupingButton := True;
    ShowGroupingEvents := True;
    Groups.Add(1,'Pro');
end
```

VFP

```
with thisform.Schedule1
.Object.Description(0) = "(todos)"
.DisplayGroupingButton = .T.
.ShowGroupingEvents = .T.
.Groups.Add(1,"Pro")
endwith
```

dBASE Plus

local oSchedule

```
oSchedule = form.Activex1.nativeObject
```

```
oSchedule.Template = [Description(0) = "(todos)"] // oSchedule.Description(0) = "
(todos)"
oSchedule.DisplayGroupingButton = true
oSchedule.ShowGroupingEvents = true
oSchedule.Groups.Add(1,"Pro")
```

XBasic (Alpha Five)

```
Dim oSchedule as P

oSchedule = topparent:CONTROL_ACTIVEX1.activex

oSchedule.Template = "Description(0) = \"(todos)\"" ' oSchedule.Description(0) =

"(todos)"

oSchedule.DisplayGroupingButton = .t.

oSchedule.ShowGroupingEvents = .t.

oSchedule.Groups.Add(1,"Pro")
```

Visual Objects

oDCOCX_Exontrol1:[Description,exGroupBarAll] := "(todos)" oDCOCX_Exontrol1:DisplayGroupingButton := true oDCOCX_Exontrol1:ShowGroupingEvents := true oDCOCX_Exontrol1:Groups:Add(1,"Pro")

PowerBuilder

```
OleObject oSchedule
```

oSchedule = ole_1.Object oSchedule.**Description**(0,"(todos)") oSchedule.DisplayGroupingButton = true oSchedule.ShowGroupingEvents = true oSchedule.Groups.Add(1,"Pro")

property Schedule.DisplayGroupingButton as Boolean

Gets or sets a value that indicates whether the grouping button is displayed in the date header.

| Туре | Description |
|---------|--|
| Booloon | A Boolean expression that specifies whether the date's |
| Doolean | header displays a grouping button. |

By default, the DisplayGroupingButton property is False. The DisplayGroupingButton property indicates whether the header of the date displays the grouping button. The list of available groups is displayed on a drop down panel, once the user clicks the grouping/filtering button. The drop down list shows the <u>Title</u> for each group found. The ShowGroupingEvents property indicates whether the control displays events grouped by its GroupID property. The ApplyGroupingColors property specifies whether the control uses the Group's <u>EventBackColor</u> / <u>EventForeColor</u> / <u>EventPattern</u> properties to show the events in the groups. The <u>SingleGroupingView</u> property specifies whether the drop down panel shows radio buttons, instead check boxes, so the user can see all groups or only one group at the time. The <u>GroupID</u> property specifies the identifier of the group where the event belongs. If the control displays groups the GroupID property of the newly created event is automatically updated with the group where the event has been created. The <u>AllowMoveEventToOtherGroup</u> property specifies whether the user can move an event from a group to another at runtime. Use the <u>Add</u> method of the Groups collection to add new groups to the control.

The grouping button is displayed if:

- DisplayGroupingButton property is True
- ShowGroupingEvents property is True

The control displays groups if:

- <u>ShowGroupingEvents</u> property is True
- The <u>Groups</u> collection has elements. By default, the Groups collection contains no Group objects.

The following <u>Background</u> properties change the visual appearance of the drop down grouping panel:

- Background(exGroupingBackColor) / Background(exGroupingForeColor) changes the background and the foreground color of the panel.
- Background(exGroupingSelBackColor) / Background(exGroupingSelForeColor) changes the background and the foreground color of the selection in the panel.
- Background(exCheckBoxState0), Background(exCheckBoxState1),

Background(exCheckBoxState2) changes the visual appearance for the control's check boxes.

• Background(exRadioButtonState0), Background(exRadioButtonState1), changes the visual appearance for the control's radio buttons.

The <u>Description</u>(exGroupBarAll) property changes the "(All)" predefined string, being displayed on the top of the drop down grouping/filtering panel.

The following screen shot shows the drop down panel, if the <u>SingleGroupingView</u> property is False (by default):



The following screen shot shows the drop down panel, if the <u>SingleGroupingView</u> property is True:



property Schedule.EditContextMenuItems as String

Specifies the control's context menu, while editing the event.

| Туре | Description |
|--------|--|
| String | A string expression that indicates the items to be shown on the edit's context menu. |

The edit's context menu is displayed if the user right clicks while editing the event. Use the EditContextMenuItems property to change the edit's context menu.

By default the EditContextMenuItems property is:

Command[id=57625][captionwidth=48][group=19](Undo[id=57643][align=1] [button=-1][captionwidth=44],Redo[id=57644][align=1][button=-1]),Edit[id=57624] [captionwidth=48][group=19](Cut[id=57635][align=1][button=-1],Copy[id=57634] [align=1][button=-1],Paste[id=57637][align=1][button=-1],Delete[id=57632][align=1] [button=-1],Select All[id=57642][align=1][button=-1]),format[sep][id=57623] [height=13],Type[id=57622][show=1][captionwidth=28][group=19](**B[id=57648]** [typ=1][align=1][show=1],*l[id=57649][typ=1][align=1][show=1],U[id=57650]* [typ=1][align=1][show=1], S[id=57651][typ=1][align=1][show=1], [sep] <u>fid=576211,Afid=577601[typ=1][align=1][spchk=-1][show=1](ID[id=57761]</u> [edittype=1][editwidth=-172],Options[id=57762][edittype=1] [editwidth=-72]),G[id=57715][typ=1][align=1][spchk=-1][show=1] <u>(Color[id=57717][edittype=518][border=0][editwidth=-72],Mode[id=57724]</u> [group=17](H[id=57725][typ=2][align=1][chk=1][show=1][grp=2],V[id=57726] [typ=2][align=1][show=1][grp=2],FD[id=57727][typ=2][align=1][show=1] <u>[grp=2],BD[id=57728][typ=2][align=1][show=1][grp=2]),Blend Triangular</u> <u>Shape[id=57729][typ=1][show=-1]),O[id=57730][typ=1][align=1][spchk=-1]</u> [show=1](Color[id=57732][edittype=518][border=0] [editwidth=-96],Width[id=57739][edittype=3][border=0][min=1][max=4][freq=1] [editwidth=-72]),S[id=57743][typ=1][align=1][spchk=-1][show=1] <u>(Color[id=57745][edittype=518][border=0][editwidth=-72],Offset[id=57752]</u> [edittype=3][border=0][min=-8][max=+8][freq=1] [editwidth=-128],Width[id=57756][edittype=3][border=0][min=2][max=+7] [freq=1][editwidth=-128])),[sep][id=57620][height=4],Align[id=57619][show=1] [captionwidth=24][height=26][group=19]([id=57618][group=19] <u>(Offset[id=57709][typ=1][chk][show=1][showdis][border=0][min=-32][max=+32]</u> [freq=4][editwidth=-96][height=24])),Color[id=57618][captionwidth=28]

```
[height=26][group=3](Fore[id=57685][typ=1][show=1][showdis][editwidth=-96]
[height=24],[sep][id=57617],Back[id=57686][typ=1][show=1][showdis]
[editwidth=-96][height=24]),Font[id=57617][captionwidth=28][height=26]
[group=3](Face[id=57701][typ=1][show=1][showdis][height=24][editwidth=-116],
[sep][id=57616],Size[id=57702][typ=1][show=1][showdis][height=24]
[editwidth=-82][min=4][max=72][freq=4]),Misc[id=57609][captionwidth=24]
[group=3](Image[id=57608](Size[id=57680][edittype=515][border=0][min=16]
[max=128][freq=16][editwidth=-128][ticklabel=value = %i ? " +value : (value =
vmax ? "+value : (value = vmin ? "+value : " ) )],Insert[id=57679]()))
```

By default, the control's context menu shows as following:



Let's say we want to remove all that grouping, and shows as a regular context menu (just remove all the [group] from the EditContextMenuItems property, and you should get something like:

| Command | • | |
|------------|---|------------|
| Edit | • | Cut |
| - format - | | Сору |
| Туре | _ | Paste |
| Align | • | Delete |
| Color | • | Select All |
| Font | • | |
| Misc | • | |

The EditContextMenuItems's syntax in BNF notation:

```
<EditContextMenuItems> ::= <ITEMS>
<ITEMS> ::= <ITEM>["("<ITEMS>")"][","<ITEMS>]
<ITEM> ::= <CAPTION>[<OPTIONS>]
<OPTIONS> ::= "["<OPTION>"]"["["<OPTIONS>"]"]
<OPTION> ::= <PROPERTY>["="<VALUE>]
<PROPERTY> ::= "img" | "himg" | "sep" | "id" | "typ" | "group" | "chk" | "button" | "align" |
```

"spchk" | "show" | "rad" | "dis" | "showdis" | "bld" | "itl" | "stk" | "und" | "bg" | "fg" | "edittype" | "edit" | "mask" | "border" | "editwidth" | "captionwidth" | "height" | "grp" | "tfi" | "ttp" | "min" | "max" | "tick" | "freq" | "ticklabel" | "small" | "large" | "spin" | "ettp" | "float"

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

- img=<VALUE>, where <VALUE> is an integer expression, that indicates the index of the icon being displayed for the item.
- himg=<VALUE>, where <VALUE> indicates the key of the picture to be displayed for the item.
- sep, specifies an separator item
- id=<VALUE>, where <VALUE> is an integer expression, that indicates the identifier of the item.
- typ=<VALUE>, where <VALUE> could be one of the following:
 - **0** for regular items,
 - 1 for items that display a check/box (chk)
 - 2 to display radio buttons (rad)
- group=<VALUE>, where <VALUE> could be a bit-or combination (+) of the following values:
 - **0** (exNoGroupPopup), No grouping is performed on the sub-menu, so the subitems are shown to a float popup,
 - **1** (exGroupPopup), Groups and displays the sub-menu items on the current item, arranged from left to right
 - **2** (exNoGroupPopupFrame), Prevents showing the frame around each grouping item.
 - **4** (exGroupPopupCenter), Shows the grouping popup aligned to the center of the current item.
 - **8** (exGroupPopupRight), Shows the grouping popup aligned to the right of the current item.
 - 16 (exGroupPopupEqualSize), Shows the items that make the group of the same size
- chk[=<VALUE>], where <VALUE> could be 0 for unchecked, or not zero for checked. The chk option makes the item to display a check box. If the <VALUE> is missing the item still displays an un-checked check box.
- button=<VALUE>, where <VALUE> could be **0** for regular or **not zero** to show the item as a button.
- align=<VALUE>, where <VALUE> could be one of the following:
 - **0** (left), to align the item's caption to the left
 - $\circ~$ 1 (center), to center the item's caption
 - $\circ~\mathbf{2}$ (right), to align the item's caption to the right
- spchk=<VALUE>, where <VALUE> could be **0** for regular or **not zero** to specify whether the item's sub menu is shown only if the item is checked.

- show=<VALUE>, where <VALUE> could be 0 for regular or not zero to specify whether the checked item shows as selected
- rad=<VALUE>, where <VALUE> could be 0 for unchecked radio button or not zero to for checked radio button. Use the grp option to define the group of radio where this button should be associated, If no group of radio buttons is required, the grp could be ignored.
- dis, specifies a disabled item
- showdis=<VALUE>, where <VALUE> could be 0 for regular or not zero to specify whether the item shows as disabled, but it is still enabled
- bld, specifies that the item appears in bold
- itl, specifies that the item appears in italics
- stk, specifies that the item appears as strikeout
- und, specifies that the item is underlined
- bg=<VALUE>, specifies the item's background color, where <VALUE> could be a RGB expression (RGB(RR,GG,BB), where RR is the red value, the GG is the green value, and the BB is the blue value), or a long expression.
- fg=<VALUE>, specifies the item's foreground color, where <VALUE> could be a RGB expression (RGB(RR,GG,BB), where RR is the red value, the GG is the green value, and the BB is the blue value), or a long expression.
- edittype=<VALUE>, associates an edit field to the item, where <VALUE> could be one of the following values:
 - **0** (exItemDisableEdit), No editor is assigned to the current item.
 - $\circ~$ 1 (<code>exItemEditText</code>), A text-box editor is assigned to the current item.
 - 2 (exItemEditMask), A masked text-box editor is assigned to the current item.
 - $\circ~$ 3 (exItemEditSlider), A slider editor is assigned to the current item.
 - 4 (exItemEditProgress), A progress editor is assigned to the current item.
 - $\circ~$ 5 (<code>exItemEditScrollBar</code>), A scrollbar editor is assigned to the current item.
 - $\circ~$ 6 (<code>exItemEditColor</code>), A color editor is assigned to the current item.
 - $\circ~\textbf{7}$ (<code>exItemEditFont</code>), A font editor is assigned to the current item.
 - 256 (exItemEditReadOnly), specifies that the item's editor is shown as disabled. This value could be combined with one of the values from 0 to 7, 512
 - 512 (exItemEditSpin), A spin editor is assigned to the current item. This value could be combined with one of the values from 0 to 7, 256
- edit=<VALUE>, specifies the caption to be shown in the item's edit field, where <VALUE> could be any string
- mask=<VALUE>, specifies the mask to be applied on a masked editor. This option is valid for exItemEditMask edit. Use the float option to allow masking floating point numbers. See <u>Masking</u> for more information about <VALUE> of the mask option. See <u>Masking Float</u> for more information about <VALUE> if the float option is used.
- border=<VALUE>, specifies the border to be shown on the item's edit field, where <VALUE> could be one of the following:
 - **0** (exEditBorderNone), No border is shown.

- **-1** (exEditBorderInset), shows an inset border
- 1 (exEditBorderSingle), shows a frame border
- editwidth=<VALUE>, specifies the width to show the edit field inside the item. where
 <VALUE> could be a long expression. A negative value indicates that the field goes to the end of the item
- captionwidth=<VALUE>, specifies the width to show the HTML caption of the item. where <VALUE> could be a long expression. A negative value indicates that no limitation is applied to the item's caption, so no truncate caption is shown
- height=<VALUE>, specifies the height to show the item, where <VALUE> could be a
 positive long expression
- grp=<VALUE>, defines the radio group. It should be used when you define more groups of radio buttons. A group of radio buttons means that only one item could be checked at one time. The rad option specifies that the item displays a radio button. Use the grp option to define the group of radio where this button should be associated, If no group of radio buttons is required, the grp could be ignored. The <VALUE> could be any long expression.
- ttp=<VALUE>, defines the item's tooltip. The <VALUE> could be any HTML string expression. The item's tooltip is shown when the user hovers the item.
- min=<VALUE>, defines the minimum value of the edit field. The <VALUE> could be any long expression, and specifies the minimum value for any slider, progress, scroll, spin, or range editor.
- max=<VALUE>, defines the maximum value of the edit field. The <VALUE> could be any long expression, and specifies the maximum value for any slider, progress, scroll, spin, or range editor.
- tick=<VALUE>, defines where the ticks of the slider edit appear. This option is valid for exItemEditSlider edit. The <VALUE> could be one of the following values:
 - **0** (exBottomRight), The ticks are displayed on the bottom/right side.
 - **1** (exTopLeft), The ticks are displayed on the top/left side.
 - **2** (exBoth), The ticks are displayed on the both side.
 - **3** (exNoTicks), No ticks are displayed.
- freq=<VALUE>, indicates the ratio of ticks on the slider edit. This option is valid for exItemEditSlider edit. The <VALUE> could be a positive long expression.
- ticklabel=<VALUE>, indicates the HTML label to be displayed on slider's ticks. This
 option is valid for exItemEditSlider edit. See <u>Tick Label Expression</u> for more information
 about <VALUE> of the ticklabel option.
- small=<VALUE>, indicates the amount by which the edit's position changes when the user presses the arrow key (left, right, or button). This option is valid for exItemEditSlider, exItemEditScrollBar edit. The <VALUE> could be a positive long expression.
- large=<VALUE>, indicates the amount by which the edit's position changes when the user presses the CTRL + arrow key (CTRL + left, CTRL + right). This option is valid for exItemEditSlider, exItemEditScrollBar edit. The <VALUE> could be a positive long expression.

- spin=<VALUE>, specifies the step to advance when user clicks the editor's spin.. This
 option is valid for exItemEditSpin edit. The <VALUE> could be a positive long
 expression.
- ettp=<VALUE>, specifies the HTML tooltip to be shown when the item's value is changed. This option is valid for exItemEditSlider/exItemEditScrollBar edit. The <VALUE> could be any string expression.
- float=<VALUE>, Specifies whether the mask field masks a floating point number. This option is valid for exItemEditMask edit. See <u>Masking Float</u> for more information about <VALUE> of mask option, if the float option is used. The <VALUE> could be **0** for standard masking field or **not zero** to specify that the field is masking a floating point.

ContextMenu - Masking

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

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

indicates the following:

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

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

Input masks are made up one mandatory part and three optional parts, and each part is

separated by a semicolon (;). If a part should use the semicolon (;) it must uses the $\$ instead

The purpose of each part is as follows:

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

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

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

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

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

The known options for the forth part are:

- float, indicates that the field is edited as a decimal number, integer. The first part of the input mask specifies the pattern to be used for grouping and decimal separators, and - if negative numbers are supported. If the first part is empty, the float is formatted as indicated by current regional settings. For instance, "##;;;float" specifies a 2 digit number in float format. The grouping, decimal, negative and digits options are valid if the float option is present.
- grouping=value, Character used to separate groups of digits to the left of the decimal. Valid only if float is present. For instance ";;;;float,grouping=" indicates that no grouping is applied to the decimal number (LOCALE_STHOUSAND)
- decimal=value, Character used for the decimal separator. Valid only if float is present. For instance ";;;float,grouping= ,decimal=\," indicates that the decimal number uses the space for grouping digits to the left, while for decimal separator the comma character is used (LOCALE_SDECIMAL)
- negative=value, indicates whether the decimal number supports negative numbers. The value should be 0 or 1. 1 means negative numbers are allowed. Else 0 or missing, the negative numbers are not accepted. Valid only if float is present.
- digits=value, indicates the max number of fractional digits placed after the decimal separator. Valid only if float is present. For instance, ";;;float,digits=4" indicates a max 4 digits after decimal separator (LOCALE_IDIGITS)
- *password*[=value], displays a black circle for any shown character. For instance,

";;;;password", specifies that the field to be displayed as a password. If the value parameter is present, the first character in the value indicates the password character to be used. By default, the * password character is used for non-TrueType fonts, else the black circle character is used. For instance, ";;;password=*", specifies that the field to be displayed as a password, and use the * for password character. If the value parameter is missing, the default password character is used.

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

and the field is keep opened while the value is inappropriate. For instance, "! (999) 000 0000;;;;invalid=The value you entered isn't appropriate for the input mask '<%mask%>' specified for this field." displays the "The value you entered isn't appropriate for the input mask '...' specified for this field." tooltip once the user leaves the field and it is not-valid (for instance, the field includes entities required and uncompleted). The <%mask%> keyword in value, substitute the current mask of the field, while the <%value%> keyword substitutes the current value (including the literals). If this option should display/use the semicolon (;) character is should be \; (escape). This option can be combined with empty, validateas.

 validateas=value, specifies the additional validation is done for the current field. If value is missing or 0 (exValidateAsNone), the option has no effect. The validateas option has effect only if the invalid option specifies a not-empty value. Currently, the value can be 1 (exValidateAsDate), which indicates that the field is validated as a date. For instance, having the mask "!00/00/0000;;0;empty,validateas=1,invalid=Invalid date!,warning=Invalid character!,select=4,overtype", indicates that the field is validate as date (

validateas=1).

• **empty**, indicates whether the field supports empty values. This option can be used with invalid flag, which indicates that the user can leave the field if it is empty. If empty flag is present, the field displays nothing if no entity is completed (empty). Once the user starts typing characters the current mask is displayed. For instance, having the mask "!(999) 000

0000;;;empty,select=4,overtype,invalid=invalid phone number,beep", it specifies an empty or valid phone to be entered.

select=value, indicates what to select from the field when it got the focus. The value could be 0 (nothing, exSelectNoGotFocus), 1 (select all, exSelectAllGotFocus), 2 (select the first empty and editable entity of the field, exSelectEditableGotFocus), 3 (moves the cursor to the beginning of the first empty and editable entity of the field, exMoveEditableGotFocus), 4 (select the first empty, required and editable entity of the field,

exSelectRequiredEditableGotFocus), 5 (moves the cursor to the beginning of the first empty, required and editable entity of the field,

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

Experimental:

multiline, specifies that the field supports multiple lines.

rich, specifies that the field displays a rich type editor. By default, the standard edit field is shown

disabled, shows as disabled the field.

ContextMenu - Masking Float

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

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

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

Here are few samples:

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

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

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

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

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

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

ContextMenu - Tick Label Expression

min 17 21 25 29 33 37 41 45 49 53 57 61 65 69 73 77 81 85 max

For instance:

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

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

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

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

The supported binary arithmetic operators are:

- * (multiplicity operator), priority 5
- *I* (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; #5/1/2002#:1)*" indicates that only *#1/1/2002#, #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
- $\circ~$ 19 unsigned long
- 20 long on 64 bits
- 21 unsigned long on 64 bites

- str (unary operator) converts the expression to a string
- dbl (unary operator) converts the expression to a number
- **date** (unary operator) converts the expression to a date, based on your regional settings
- **dateS** (unary operator) converts the string expression to a date using the format MM/DD/YYYY HH:MM:SS.

Other known operators for numbers are:

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

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

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

field "Negative number format" from "Regional and Language Options" is using. The valid values are 0, 1, 2, 3 and 4 with the following meanings:

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

Other known operators for strings are:

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

Other known operators for dates are:

- **time** (unary operator) retrieves the time of the date in string format, as specified in the control's panel.
- **timeF** (unary operator) retrieves the time of the date in string format, as "HH:MM:SS". For instance the timeF(1:23 PM) returns "13:23:00"
- **shortdate** (unary operator) formats a date as a date string using the short date format, as specified in the control's panel.

- shortdateF (unary operator) formats a date as a date string using the "MM/DD/YYYY" format. For instance the shortdateF(December 31, 1971 11:00 AM) returns "12/31/1971".
- **dateF** (unary operator) converts the date expression to a string expression in "MM/DD/YYYY HH:MM:SS" format.
- **longdate** (unary operator) formats a date as a date string using the long date format, as specified in the control's panel.
- year (unary operator) retrieves the year of the date (100,...,9999)
- month (unary operator) retrieves the month of the date (1, 2,...,12)
- day (unary operator) retrieves the day of the date (1, 2,...,31)
- **yearday** (unary operator) retrieves the number of the day in the year, or the days since January 1st (0, 1,...,365)
- weekday (unary operator) retrieves the number of days since Sunday (0 Sunday, 1 Monday,..., 6 Saturday)
- hour (unary operator) retrieves the hour of the date (0, 1, ..., 23)
- min (unary operator) retrieves the minute of the date (0, 1, ..., 59)
- sec (unary operator) retrieves the second of the date (0, 1, ..., 59)

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

- displays the text in **bold**.
- <i></i> displays the text in *italics*.
- <u></u> <u>underlines</u> the text.
- <s></s> Strike-through text
- 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> displays text with a specified foreground color. The RR, GG or BB should be hexa values and indicates red, green and blue values.
- <bgcolor=RRGGBB></bgcolor> displays text with a specified background color. The RR, GG or BB should be hexa values and indicates red, green and blue values.
- **
>** a forced line-break
- <**solidline**> The next line shows a solid-line on top/bottom side. If has no effect for a single line caption.
- **<dotline>** The next line shows a dot-line on top/bottom side. If has no effect for a single line caption.
- **<upline>** The next line shows a solid/dot-line on top side. If has no effect for a single line caption.
- <r> Right aligns the text
- **<c>** Centers the text
- number[:width] 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.
- key[:width] 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, in UNICODE configuration. The & ampersand is only recognized as markup when it is followed by a known letter or a # character and a digit. For instance if you want to display bold in HTML caption you can use bold

property Schedule. Enabled as Boolean

Enables or disables the control.

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

By default, the Enabled property is True. Use the Enabled property to disable the control. The <u>DisableZoneFormat</u> property of the Calendar returns or sets an expression that determines the dates being disabled in the calendar/schedule panel. The <u>AllowUpdateDisableZone</u> property on True, lets user to update the disabled zones. The <u>AllowCreateEvent</u> property indicates the combination of the keys to let user creates new events. The <u>AllowMoveEvent</u> property indicates the combination of the keys to let user moves the events. The <u>AllowResizeEvent</u> property indicates the combination of the keys to let user let user resizes the events.

method Schedule.EndBlockUndoRedo ()

Ends recording the UI operations and adds the undo/redo operations as a block, so they all can be restored at once, if Undo method is performed.

Туре

Description

You can use the <u>StartBlockUndoRedo</u> / EndBlockUndoRedo methods to group multiple Undo/Redo operations into a single-block. The GroupUndoRedoActions groups the next to current Undo/Redo Actions in a single block. A block may hold multiple Undo/Redo actions. The <u>AllowUndoRedo</u> property enables or disables the Undo/Redo feature. Use the GroupUndoRedoActions method to group two or more entries in the Undo/Redo queue in a single block, so when a next Undo/Redo operation is performed, multiple actions may occur. For instance, moving several calendar-events in the same time (multiple calendar-events selection) is already recorded as a single block. Use the <u>UndoRedoQueueLength</u> property to specify the number of entries that Undo/Redo queue may store.

A block starts with StartBlock and ends with EndBlock when listed by <u>UndoListAction</u>/<u>RedoListAction</u> property as in the following sample:

StartBlock MoveEvent;B MoveEvent;A EndBlock

method Schedule.EndUpdate ()

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

Туре

Description

The <u>BeginUpdate</u> and EndUpdate methods increases the speed of loading your events, by preventing painting the control when it suffers any change. Once that BeginUpdate method was called, you have to make sure that EndUpdate method will be called too. You can use the <u>Refresh</u> method to refresh the control's content.

method Schedule.EnsureVisible (Date as Variant)

Ensures that the specified date fits the client area of the schedule view.

Туре	Description
Date as Variant	A DATE expression that specifies the date to be shown in the schedule view.

The EnsureVisible method ensures that giving date fits the schedule's view. The EnsureVisible method does not select a date, and it scrolls the schedule view so it ensures that the panel shows the specified date. You can use the <u>OnSelectDate</u> property to specifiy the action to perform once the user clicks or selects a date in the calendar panel. The <u>DayViewWidth</u> property specifies the width, in pixels, of the date in the schedule panel. The <u>DayViewOffsetX</u> property indicates the horizontal scroll position of the schedule's view. The <u>DayViewOffsetY</u> property specifies the wight, in pixels, of the date in the schedule panel. The <u>DayViewOffsetY</u> property indicates the vertical scroll position of the schedule's view.

property Schedule.EventFromPoint (X as OLE_XPOS_PIXELS, Y as OLE_YPOS_PIXELS) as Event

Gets the Event object from the cursor, in the schedule panel.

Туре	Description
X as OLE_XPOS_PIXELS	A single that specifies the current X location of the mouse pointer. The x values is always expressed in client coordinates1 indicates the current cursor position.
Y as OLE_YPOS_PIXELS	A single that specifies the current X location of the mouse pointer. The x values is always expressed in client coordinates1 indicates the current cursor position.
<u>Event</u>	An Event object from the position or Nothing, if not event found.

The EventFromPoint method gets the Event object from the giving position. The **EventFromPoint(-1,-1)** gives the Event object from the current mouse position. The EventFromPoint method gets nothing, if no event is found at specified location. The /NET or /WPF version provides the *EventFromPoint property and get_EventFromPoint(x,y) method* that gives the Event from the current mouse position, or from the giving (x,y) position.

The following VB sample displays the event from the cursor, if any:

```
Private Sub Schedule1_Click()

Dim e As EXSCHEDULELibCtl.Event

With Schedule1

Set e = .EventFromPoint(-1, -1)

If Not e Is Nothing Then

MsgBox e.Start & " " & e.End

End If

End With

End Sub
```

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

For instance, you can use the (get_)<u>EventFromPoint</u>(-1,-1) method during the <u>LayoutStartChanging</u>(exScheduleEditEvent) to store the Event object from the cursor to a global member, and when the <u>LayoutEndChanging</u>(exScheduleEditEvent) occurs, you can

use the previously stored member to identify the Event being edited like in the following snippet of code:

Private evEdit As Object

```
Private Sub Schedule1_LayoutStartChanging(ByVal Operation As
EXSCHEDULELibCtl.LayoutChangingEnum)
If (Operation = exScheduleEditEvent) Then
Set evEdit = Schedule1.EventFromPoint(-1, -1)
End If
End Sub
Private Sub Schedule1_LayoutEndChanging(ByVal Operation As
EXSCHEDULELibCtl.LayoutChangingEnum)
If (Operation = exScheduleEditEvent) Then
If Not evEdit Is Nothing Then
Debug.Print "Event: " & evEdit.Handle & " has been edited, and the new caption is:
" & evEdit.ExtraLabel
End If
End If
End If
```

Most of the UI parts of the control can be accessed through the mouse position as listed:

- Calendar. <u>DateFromPoint</u> property gets the date from the cursor in the Calendar panel.
- AnchorFromPoint property gets the anchor (<a>) element from the cursor.
- <u>DateTimeFromPoint</u> property gets the date/time from the cursor, in the Schedule panel.
- <u>TimeFromPoint</u> property gets the time from the cursor, in the Schedule panel.
- <u>GroupFromPoint</u> property retrieves the <u>Group</u> object from the cursor. Use the GroupHeaderFromPoint property to the Group object when cursor hovers the group's header.
- <u>GroupHeaderFromPoint</u> property retrieves the <u>Group</u> object from the cursor, when the cursor hovers just the group's header.
- <u>MarkTimeFromPoint</u> property gets the <u>MarkTime</u> object from the cursor. A MarkTime object is identified by a date/time where it is displaying.
- <u>MarkZoneFromPoint</u> property gets the <u>MarkZone</u> object from the cursor. A MarkZone object is identified by a starting date/time and ending date/time where it is displaying.
- <u>NonworkingTimeFromPoint</u> property indicates the <u>NonworkingTime</u> object from the cursor.
- <u>PictureFromPoint</u> property indicates the key of the picture from the cursor.
- <u>TimeScaleFromPoint</u> property returns the <u>TimeScale</u> object from the cursor.

property Schedule.EventParam(Parameter as Long) as Variant

Retrieves or sets a value that indicates the current's event parameter.

Туре	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 Schedule. Events as Events

Gets the Events collection of the scheduler.

Туре	Description
<u>Events</u>	An Events collection that holds the events or appointments of the control.

The <u>Add</u> method adds programmatically a new event/appointment to the control. The <u>AddEvent</u> event occurs once a new event is added to the Events collection. The <u>RemoveEvent</u> event occurs once an event is removed. The <u>UpdateEvent</u> event occurs once the margins of the event are updated. The <u>LayoutStartCreating</u>(exScheduleCreateEvent) event occurs once a new event is creating using the mouse. The <u>LayoutEndCreating</u>(exScheduleCreateEvent) event occurs once the event has been created at runtime using the mouse.

The <u>AllowCreateEvent</u> property indicates the combination of keys that user can use to create a new event at runtime. The <u>CreateEventLabel</u> property specifies the label to be shown when the user creates a new event. The <u>CreateEventLabelAlign</u> property indicates the alignment of the label while user creates new events. The

Background(exScheduleCreateEventBackColor) and

Background(exScheduleCreateEventForeColor) properties indicate the visual aspect of the label being shown to create new events.

property Schedule.EventsFont as IFontDisp

Retrieves or sets the font to display the events in the schedule view.

Туре	Description
IFontDisp	A Font object used to paint the captions/labels in the control.

The EventsFont property indicates the font to display the labels and captions on events. The ForeColor property indicates the foreground color to show the captions or labels.

The control supports the following Font properties:

- Font property, that specifies the control's font, including the Calendar's font
- EventsFont property, indicates the font to show the captions and labels on Event/Appointment objects
- <u>TimeScaleFont</u> property, specifies the font to display the labels on the control's time scales
- <u>ToolTipFont</u> property specifies the font to display the tooltip being shown when the cursor is hovering an part of the control.

property Schedule.EventsTransparent as Long

Specifies the percent of transparency to show the events in the schedule panel.

Туре	Description
Long	A long expression between 0 (opaque) and 100 (transparent) which indicates the percent of transparency to show the events in the schedule panel. For instance, the EventsTransparent property on 50 indicates that the events are shown using a semi-transparent color.

By default, the EventsTransparent property is 0. The EventsTransparent property indicates the transparency to show the events on the schedule view. The <u>ShowEventPictures</u> property specifies whether the labels are shown on the event's body. The <u>ShowEventLabels</u> property specifies whether the labels are shown on the event's body. The <u>ShowEventLabels</u> property specifies what events the control should show.

method Schedule.ExecuteTemplate (Template as String)

Executes a template and returns the result.

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

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

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

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

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

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

An instruction can be one of the following:

• **Dim** list of variables *Declares the variables. Multiple variables are separated by*

commas. (Sample: Dim h, h1, h2)

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

The Template supports the following general functions:

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

method Schedule.FitSelToView ()

Fits the selected dates to the current view.

Туре

```
Description
```

The FitSelToView method restores the view to fit the selected dates. Use the <u>Selection/SelectDate</u> property to change programmatically the dates being selected in the calendar, including the dates to be shown in the schedule view. You can use the <u>Selection/SelCount/SelDate</u> property to retrieve the selected dates. The <u>ClipToSel</u> property indicates whether the control clips the schedule panel to view the selected dates only.

The following sample shows how you can ensure that the schedule view fits the selected dates, when exCalendarAutoHide option is used on <u>OnResizeControl</u> property

Private Sub LayoutEndChanging(ByVal Operation As EXSCHEDULELibCtl.LayoutChangingEnum) If (Operation = exLayoutCalendarAutoHide) Then Schedule1.FitSelToView End If End Sub

The sample calls the FitSelToView method once the control shows or hides the calendar panel.

property Schedule.Font as IFontDisp

Retrieves or sets the control's font.

Туре	Description
IFontDisp	A Font object used to paint the captions/labels in the control.

Use the Font property to change the control's font. The ForeColor property indicates the foreground color to show the captions or labels.

The control supports the following Font properties:

- Font property, that specifies the control's font, including the Calendar's font
- <u>EventsFont</u> property, indicates the font to show the captions and labels on Event/Appointment objects
- <u>TimeScaleFont</u> property, specifies the font to display the labels on the control's time scales
- <u>ToolTipFont</u> property specifies the font to display the tooltip being shown when the cursor is hovering an part of the control.

property Schedule.ForeColor as Color

Specifies the control's foreground color.

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

The <u>Background</u>(exCalendarForeColor) property changes the calendar's panel foreground color. If this option is not set, the control's ForeColor property indicates the calendar's foreground color. The <u>BackColor</u> property specifies the control's background color. Use the <u>Picture</u> property of the control to show a picture on the control's background. The <u>Font</u> property indicates the font to show captions in the control. The <u>Font</u> property indicates the font to show captions in the control. The <u>EventsFont</u> property indicates the font to show captions in the events/appointments. The <u>BodyEventForeColor</u> property specifies the foreground color to show the body for all events. The <u>EventForeColor</u> property specifies the event's foreground color if it belongs to a group.

property Schedule.FormatAnchor(New as Boolean) as String

Specifies the visual effect for anchor elements in HTML captions.

Туре	Description
New as Boolean	A Boolean expression that indicates whether to specify the anchors never clicked or anchors being clicked.
String	A String expression that indicates the HTML format to apply to anchor elements.

By default, the FormatAnchor(True) property is "<u><fgcolor=0000FF>#" that indicates that the anchor elements (that were never clicked) are underlined and shown in light blue. Also, the FormatAnchor(False) property is "<u><fgcolor=000080>#" that indicates that the anchor elements are underlined and shown in dark blue. The visual effect is applied to the anchor elements, if the FormatAnchor property is not empty. For instance, if you want to do not show with a new effect the clicked anchor elements, you can use the FormatAnchor(False) = "", that means that the clicked or not-clicked anchors are shown with the same effect that's specified by FormatAnchor(True). An anchor is a piece of text or some other object (for example an image) which marks the beginning and/or the end of a hypertext link. The <a> element is used to mark that piece of text (or inline image), and to give its hypertextual relationship to other documents. The control fires the AnchorClick event to notify that the user clicks an anchor element. This event is fired only if prior clicking the control it shows the hand cursor. The AnchorClick event carries the identifier of the anchor, as well as application options that you can specify in the anchor element. The hand cursor is shown when the user hovers the mouse on the anchor elements.

The control fires the <u>AnchorClick</u> event once the user clicks an anchor element. You can use the <u>AnchorFromPoint</u> property to retrieve the identifier of the anchor element from the cursor.

property Schedule.GroupFromPoint (X as OLE_XPOS_PIXELS, Y as OLE_YPOS_PIXELS) as Group

Retrieves the Group object from the cursor, in the schedule panel.

Туре	Description
X as OLE_XPOS_PIXELS	A single that specifies the current X location of the mouse pointer. The x values is always expressed in client coordinates.
Y as OLE_YPOS_PIXELS	A single that specifies the current Y location of the mouse pointer. The y values is always expressed in client coordinates.
<u>Group</u>	A Group object from the cursor.

The <u>MouseMove</u> event is generated continually as the mouse pointer moves across objects. During the MouseMove event you can call the <u>ShowToolTip</u> method to display any custom tooltip. During the <u>Click</u> or <u>RClick</u> event you can get an UI part of the control using one of the following properties. *All ...FromPoint properties can be use such as ...FromPoint(-1,-1) to get the UI part of the control from the current mouse position, in other words, you do not have to pass any X, Y coordinates.*

You can get UI parts from the cursor, using any of the following ... FromPoint properties:

- Calendar. <u>DateFromPoint</u> property gets the date from the cursor in the Calendar panel.
- AnchorFromPoint property gets the anchor (<a>) element from the cursor.
- <u>DateTimeFromPoint</u> property gets the date/time from the cursor, in the Schedule panel.
- <u>TimeFromPoint</u> property gets the time from the cursor, in the Schedule panel.
- EventFromPoint property gets the Event object from the cursor.
- <u>GroupFromPoint</u> property retrieves the <u>Group</u> object from the cursor. Use the GroupHeaderFromPoint property to the Group object when cursor hovers the group's header.
- <u>GroupHeaderFromPoint</u> property retrieves the <u>Group</u> object from the cursor, when the cursor hovers just the group's header.
- <u>MarkTimeFromPoint</u> property gets the <u>MarkTime</u> object from the cursor. A MarkTime object is identified by a date/time where it is displaying.
- <u>MarkZoneFromPoint</u> property gets the <u>MarkZone</u> object from the cursor. A MarkZone object is identified by a starting date/time and ending date/time where it is displaying.
- <u>NonworkingTimeFromPoint</u> property indicates the <u>NonworkingTime</u> object from the cursor.
- <u>PictureFromPoint</u> property indicates the key of the picture from the cursor.
- <u>TimeScaleFromPoint</u> property returns the <u>TimeScale</u> object from the cursor.

All ... FromPoint properties that returns an Object, may return Nothing, Empty or NULL, if no

object is found.

property Schedule.GroupHeaderFromPoint (X as OLE_XPOS_PIXELS, Y as OLE_YPOS_PIXELS) as Group

Retrieves the Group's header from the cursor, in the schedule panel.

Туре	Description
X as OLE_XPOS_PIXELS	A single that specifies the current X location of the mouse pointer. The x values is always expressed in client coordinates.
Y as OLE_YPOS_PIXELS	A single that specifies the current Y location of the mouse pointer. The y values is always expressed in client coordinates.
<u>Group</u>	A Group object from the cursor. The GroupHeaderFromPoint property returns an object only if it hovers the group's header.

The <u>MouseMove</u> event is generated continually as the mouse pointer moves across objects. During the MouseMove event you can call the <u>ShowToolTip</u> method to display any custom tooltip. During the <u>Click</u> or <u>RClick</u> event you can get an UI part of the control using one of the following properties. *All ...FromPoint properties can be use such as ...FromPoint(-1,-1) to get the UI part of the control from the current mouse position, in other words, you do not have to pass any X, Y coordinates.*

You can get UI parts from the cursor, using any of the following ... FromPoint properties:

- Calendar. <u>DateFromPoint</u> property gets the date from the cursor in the Calendar panel.
- <u>AnchorFromPoint</u> property gets the anchor (<a>) element from the cursor.
- <u>DateTimeFromPoint</u> property gets the date/time from the cursor, in the Schedule panel.
- <u>TimeFromPoint</u> property gets the time from the cursor, in the Schedule panel.
- EventFromPoint property gets the Event object from the cursor.
- <u>GroupFromPoint</u> property retrieves the <u>Group</u> object from the cursor. Use the GroupHeaderFromPoint property to the Group object when cursor hovers the group's header.
- <u>GroupHeaderFromPoint</u> property retrieves the <u>Group</u> object from the cursor, when the cursor hovers just the group's header.
- <u>MarkTimeFromPoint</u> property gets the <u>MarkTime</u> object from the cursor. A MarkTime object is identified by a date/time where it is displaying.
- <u>MarkZoneFromPoint</u> property gets the <u>MarkZone</u> object from the cursor. A MarkZone object is identified by a starting date/time and ending date/time where it is displaying.
- <u>NonworkingTimeFromPoint</u> property indicates the <u>NonworkingTime</u> object from the cursor.
- <u>PictureFromPoint</u> property indicates the key of the picture from the cursor.

• <u>TimeScaleFromPoint</u> property returns the <u>TimeScale</u> object from the cursor.

All ... FromPoint properties that returns an Object, may return Nothing, Empty or NULL, if no object is found.

property Schedule.GroupHighlightEvent as Boolean

Highlights the date in the schedule panel using the HighlightEvent property of each Group found on day's events.

Туре	Description
Boolean	A Boolean expression that specifies whether the dates in the schedule panel using the ScheduleHighlightEvent property of each Group found on day's events

By default, the GroupHighlightEvent property is False. The GroupHighlightEvent property specifies if events are highlighted using the <u>HighlightEvent</u> property (False), or using the <u>CalendarHighlightEvent</u> property of the <u>Group</u> that event belongs to (True). The <u>ShowHighlightEvent</u> property specifies whether the calendar panel highlights the events in the calendar panel. The <u>ScheduleHighlightEvent</u> property specifies the visual appearance of dates with events in the schedule panel. The <u>GroupID</u> property indicates the identifier of the event's group.

property Schedule.Groups as Groups

Retrieves the Groups collection of the scheduler.

Туре	Description
<u>Groups</u>	A Groups object that holds the Group objects.

The Groups collection holds the groups in the control. Use the <u>Add</u> method of the Groups collection to add new groups to the control. The <u>DisplayGroupingButton</u> property indicates whether the header of the date displays the grouping button. The list of available groups is displayed on a drop down panel, once the user clicks the grouping/filtering button. The drop down list shows the <u>Title</u> for each group found. The <u>ShowGroupingEvents</u> property indicates whether the control displays events grouped by its <u>GroupID</u> property. The <u>ApplyGroupingColors</u> property specifies whether the control uses the Group's <u>EventBackColor / EventForeColor / EventPattern</u> properties to show the events in the groups. The <u>SingleGroupingView</u> property specifies whether the drop down panel shows radio buttons, instead check boxes, so the user can see all groups or only one group at the time. The <u>GroupID</u> property specifies the identifier of the group where the event belongs. If the control displays groups the GroupID property of the newly created event is automatically updated with the group where the event has been created. The <u>AllowMoveEventToOtherGroup</u> property specifies whether the user can move an event from a group to another at runtime.

The control displays groups if:

- <u>ShowGroupingEvents</u> property is True
- The <u>Groups</u> collection has elements. By default, the Groups collection contains no Group objects.

The following <u>Background</u> properties change the visual appearance of the drop down grouping panel:

- Background(exGroupingBackColor) / Background(exGroupingForeColor) changes the background and the foreground color of the panel.
- Background(exGroupingSelBackColor) / Background(exGroupingSelForeColor) changes the background and the foreground color of the selection in the panel.
- Background(exCheckBoxState0), Background(exCheckBoxState1), Background(exCheckBoxState2) changes the visual appearance for the control's check boxes.
- Background(exRadioButtonState0), Background(exRadioButtonState1), changes the visual appearance for the control's radio buttons.

The <u>Description</u>(exGroupBarAll) property changes the "(All)" predefined string, being displayed on the top of the drop down grouping/filtering panel.

The following screen shot shows the events with grouping colors applied, including the grouping panel:



The following screen shot shows the events with grouping colors applied, but with no grouping button/panel:



The following screen shot shows the events with no grouping colors apply:



method Schedule.GroupUndoRedoActions (Count as Long)

Groups the next to current Undo/Redo Actions in a single block.

Туре	Description
Count as Long	A Long expression that specifies the number of entries being grouped in a single block of actions, in the Undo/Redo queue.

The GroupUndoRedoActions groups the next to current Undo/Redo Actions in a single block. A block may hold multiple Undo/Redo actions. The <u>AllowUndoRedo</u> property enables or disables the Undo/Redo feature. Use the GroupUndoRedoActions method to group two or more entries in the Undo/Redo queue in a single block, so when a next Undo/Redo operation is performed, multiple actions may occur. You can use the <u>StartBlockUndoRedo</u> / <u>EndBlockUndoRedo</u> methods to group multiple Undo/Redo operations into a single-block. For instance, moving several calendar-events in the same time (multiple calendar-events selection) is already recorded as a single block. Use the <u>UndoRedoQueueLength</u> property to specify the number of entries that Undo/Redo queue may store.

A block starts with StartBlock and ends with EndBlock when listed by <u>UndoListAction</u>/<u>RedoListAction</u> property as in the following sample:

StartBlock MoveEvent;B MoveEvent;A EndBlock

property Schedule.HeaderAllDayEventHeight as Double

Specifies the height of the All-Day events being displayed on the control's All-Day header.

Туре	Description
Double	A double expression that indicates the height of the event to be shown in All-Day header. A positive value is multiplied with the font's height, while a negative value indicates a fixed height. For instance, 1 indicates that's the event's height is indicated by the control's font, while -20 indicates that the event's height is 20 pixels.

By default, the HeaderAllDayEventHeight property is 1, which indicates that the height of the event in the All-Day header is the same as the font's height. The HeaderAllDayEventHeight property specifies the height of the events to be displayed in the All-Day header. Use the <u>ShowAllDayHeader</u> property to show the All-Day header, so all All-Day events are displayed on this header instead in the time scale section of the schedule view. The <u>AllDayEvent</u> property indicates whether the event is an All-Day event. Use the <u>Background</u>(exScheduleEventContinuePrevWeek)/<u>Background</u>(exScheduleEventContinueNe: property to specify an EBN object to be shown on the start/end of the week, if the event continues on multiple days.

The following screen shot shows the events in the All-Day header with the following EBN files:

- ____ / <u>gtask.ebn</u>, specifies the visual appearance for the event's body
- gprev.ebn, indicates the arrow to be shown on the start of the week, if the event continues on previously week.
- I gnext.ebn, indicates the arrow to be shown on the end of the week, if the event continues on the next week.

		May1	2	3	4	5
		5/1/2012 - 5/9/2012				
			5/2/2012 - 5/12/2012			
6	7	8	9	10	11	12
5/1/2012 - 5/9/2012						
5/2/2012 - 5/12/2012						
13	14	15	16	17	18	19
	5/14/2012 - 5/24/2012					
		5/15/2012 - 5/23/2012				
20		1000	225			
20	21	22	23	24	25	26
5/14/2012 - 5/24/2012	21	22	23	24	25 5/25/2012 - 5/30/2012	26
5/14/2012 - 5/24/2012 5/15/2012 - 5/23/2012	21	22	23	24	25 5/25/2012 - 5/30/2012	26 5/28/2012 - 5/30/2012
5/14/2012 - 5/24/2012 5/15/2012 - 5/23/2012	21	22	23	24	25 5/25/2012 - 5/30/2012	26 5/28/2012 - 5/30/2012
20 5/14/2012 - 5/24/2012 5/15/2012 - 5/23/2012 27	21	22	23	24	25 5/25/2012 - 5/30/2012	26 5/26/2012 - 5/30/2012
20 5/14/2012 - 5/24/2012 5/15/2012 - 5/23/2012 27 5/25/2012 - 5/30/2012	21	22	30	24	25 5/25/2012 - 5/30/2012	26 5/28/2012 - 5/30/2012
20 5/14/2012 - 5/24/2012 5/15/2012 - 5/23/2012 27 5/25/2012 - 5/30/2012 5/28/2012 - 5/30/2012	21	22	23	24	25 5/25/2012 - 5/30/2012	26 5/28/2012 - 5/30/2012

The following screen shot shows the events in the All-Day header with no EBN files:

		May1	2	3	4	5
		5/1/2012 - 5/9/2012				,
			5/2/2012 - 5/12/2012			•
6	7	8	9	10	11	12
5/1/2012 - 5/9/2012						
5/2/2012 - 5/12/2012						
13	14	15	16	17	18	19
	5/14/2012 - 5/24/2012					,
		5/15/2012 - 5/23/2012				
20	21	22	23	24	25	26
5/14/2012 - 5/24/2012					5/25/2012 - 5/30/2012	•
• 5/15/2012 - 5/23/2012						5/28/2012 - 5/30/2012
27	28	29	30	May31	2	
5/25/2012 - 5/30/2012						
5/26/2012 - 5/30/2012					-	

The following template x-script (<u>exhelper</u>), shows how to assign EBN appearance to events, including the next, prev signs:

```
BeginUpdate
OnResizeControl = 2048
ShowAllDayHeader = True
Calendar
{
  SelectDate(#5/8/2012#) = True
  Select(3)
  SelectDate(#5/15/2012#) = False
  Select(19)
}
VisualAppearance.Add(1,"E:\Exontrol\ExG2antt\sample\EBN\gtask.ebn")
VisualAppearance.Add(2,"E:\Exontrol\ExG2antt\sample\EBN\gnext.ebn")
VisualAppearance.Add(3,"E:\Exontrol\ExG2antt\sample\EBN\gprev.ebn")
BodyEventBackColor = 0x100000
Background(86) = 0x2000000
Background(85) = 0x3000000
HeaderAllDayEventHeight = -20
Events.Add(#5/8/2012#,#5/17/2012#).AllDayEvent = True
EndUpdate()
```

property Schedule.HeaderDayHeight as Double

Indicates the height of the day's header.

Туре	Description
Double	A double expression that indicates the height of the header to be shown the date's labels, in the schedule view. A positive value is multiplied with the font's height, while a negative value indicates a fixed height.

By default, the HeaderDayHeight property is 1.5. The HeaderDayHeight property determines the height of the header by multiplying the value with the height of the current font, if the value is positive, else if the value is negative it indicates a fixed height. The Font property indicates the control's font. Use the HeaderDayHeight property to programmatically extend the header's height. The HeaderDayHeight property on 0, indicates that no header is shown for dates. The HeaderDayShortLabel property indicates the labels to be displayed on the date's header when the schedule vies is minimized so no time scale is shown, while the HeaderDayLongLabel property indicates the labels to be shown on date's header when the schedule view displays time scales.

The <u>HeaderGroupHeight</u> property indicates the height of the header that displays groups in the control.

property Schedule.HeaderDayLongLabel as String

Specifies the long label for header days.

Туре	Description
String	A String expression that defines alternative HTML labels to be shown on the date's header.

By default, the HeaderDayLongLabel property is "<|><%dddd%>, <%mmm%> <%d%>, <%yyyy%><|><%dddd%>,<%mmm%> <%d%>,<%yyyy%><|><%dddd%>, <%mmm%> <%d%>,`<%yy%><|><%dddd%>, <%mmm%> <%d%><|><%dddd%>, <%m3%> <%d%><|><%dddd%>, <%mmm%> <%d3%><|><%dd2%><|> <%d1%>". The HeaderDayLongLabel property defines the alternate HTML labels being shown on date's header, when the time scale is visible. The <u>HeaderDayShortLabel</u> property defines the HTML label to show on date's header when no time scale is available (the schedule view has been shrink until the time scale is hidden). The <u>HeaderDayHeight</u> property indicates the height of the day's header. The <u>HeaderDayLabel</u> property specifies the HTML date-format to be shown on the calendar's header.

The following screen shot shows the header's schedule view if using "<|><sha><%dddd%>, <%mmm%> <%d%>, <fgcolor=FF0000><%loc_yyyy%><|><sha><%dddd%>, <%mmm%> <%d%>,<fgcolor=FF0000><%loc_yyy%><|><sha><%dddd%>, <%mmm%> <%d%>,`<fgcolor=FF0000><%loc_yy%><|><sha><%dddd%>, <%mmm%> <%d%><|><sha><%dddd%>, <%mmm%> <%d%><|><sha><%dddd%>, <%d%><|><sha><%dddd%>, <%d%><|><sha><%dddd%>,

-	งันทร์, สีงหาคม27, <mark>2555</mark>	อังการ, สิงหากม28, 2555	พธ, สีงหาคม29, 2555	T
	1/8/2555 - 31/8/2555			4
	,	28/8/2555 - 8/9/2555		Ŧ
11:00				
:15			11:15 AM - 2:45 PM	
:45				
12:00				-

The following screen shot shows the header's schedule view if using "<|><%dddd%>,

<%mmm%> <%d%>, <fgcolor=FF0000><%loc_yyyy%><|><%dddd%>,<%mmm%> <%d%>,<fgcolor=FF0000><%loc_yyy%><|><%dddd%>, <%mmm%> <%d%>,`<fgcolor=FF0000><%loc_yy%><|><%dddd%>, <%mmm%> <%d%><|> <%dddd%>, <%m3%> <%d%><|><%dddd%>, <%dddd%><|><%dddd%><|><%dddd%><|> <%d2%><|><%d1%>" (default)

	อันทร์, สิงหาคม 27, <mark>2555</mark>	มังการ, สิงหาคม 28, 2555	พุธ, สิงหาคม 29, <mark>2555</mark>
1/8/25	65 - 31/8/2555		
		28/8/2555 - 8/9/2555	
11:00			
:15			11:15 AM - 2:45 PM
:30			
12:00			

The format of the HeaderDayLongLabel property is "ALT1[<|>ALT2<|>...]" where

• ALT defines a HTML label that supports a lot of predefined built-in tags.

For instance, HeaderDayLongLabel could be

- "<%d3%>", always displays a 3-letters from the week day.
- "<|><%d1%><|><%d2%><|><%d3%><|><%dddd%>", may display nothing, 1, 2, 3 letters, or the full week day.
- "<|><%d1%><|><%d2%><|><%d3%><|><%dddd%><|><%d3%>, <%m3%>
 <%d%>, '<%yy%><|><%dddd%>, <%mmm%> <%d%>, <%yyyy%>" indicates a list of 7 alternate HTML labels that the control uses when it displays the date header.

When the user resizes the schedule view, the control searches for the best fit, and the date header displays the proper HTML label, based on its width, so no ... (three dots) is shown in the labels as much as possible.

The HeaderDayLongLabel property supports the following built-in tags:

- <%d%> Day of the month in one or two numeric digits, as needed (1 to 31).
- <%dd%> Day of the month in two numeric digits (01 to 31).
- <%d1%> First letter of the weekday (S to S). (Use the <u>WeekDays</u> property to specify the name of the days in the week)
- <%loc_d1%> Indicates day of week as a one-letter abbreviation using the current user settings.
- <%d2%> First two letters of the weekday (Su to Sa). (Use the <u>WeekDays</u> property to specify the name of the days in the week)
- <%loc_d2%> Indicates day of week as a two-letters abbreviation using the current user settings.
- <%d3%> First three letters of the weekday (Sun to Sat). (Use the <u>WeekDays</u> property to specify the name of the days in the week)
- <%loc_d3%> equivalent with <%loc_ddd%>
- <%ddd%> First three letters of the weekday (Sun to Sat). (Use the <u>WeekDays</u> property to specify the name of the days in the week). You can use the <%loc_ddd%> that indicates the day of week as a three-letter abbreviation using the current user regional and language settings.

- <%loc_ddd%> Indicates the day of week as a three-letter abbreviation using the current user regional and language settings.
- <%dddd%> Full name of the weekday (Sunday to Saturday). (Use the <u>WeekDays</u> property to specify the name of the days in the week). You can use the <%loc_dddd%> that indicates day of week as its full name using the current user regional and language settings.
- <%loc_dddd%> Indicates day of week as its full name using the current user regional and language settings.
- <%i%> Displays the number instead the date. For instance, you can display numbers as 1000, 1001, 1002, 1003, instead dates. (the valid range is from -647,434 to 2,958,465)
- <%**w**%> Day of the week (1 to 7).
- <%**ww**%> Week of the year (1 to 53).
- <%m%> Month of the year in one or two numeric digits, as needed (1 to 12).
- <%mr%> Month of the year in Roman numerals, as needed (I to XII).
- <%mm%> Month of the year in two numeric digits (01 to 12).
- <%m1%> First letter of the month (J to D). (Use the <u>MonthNames</u> property to specify the name of the months in the year)
- <%loc_m1%> Indicates month as a one-letter abbreviation using the current user settings.
- <%m2%> First two letters of the month (Ja to De). (Use the <u>MonthNames</u> property to specify the name of the months in the year)
- <%loc_m2%> Indicates month as a two-letters abbreviation using the current user settings.
- <%m3%> First three letters of the month (Jan to Dec). (Use the <u>MonthNames</u> property to specify the name of the months in the year)
- <%loc_m3%> equivalent with <%loc_mmm%>
- <%mmm%> First three letters of the month (Jan to Dec). (Use the MonthNames property to specify the name of the months in the year). You can use the <%loc_mmm%> that indicates month as a three-letter abbreviation using the current user regional and language settings.
- <%loc_mmm%> Indicates month as a three-letter abbreviation using the current user regional and language settings.
- <%mmm%> Full name of the month (January to December). (Use the <u>MonthNames</u> property to specify the name of the months in the year). You can use the <%loc_mmm%> that indicates month as its full name using the current user regional and language settings.
- <%loc_mmmm%> Indicates month as its full name using the current user regional and language settings.
- <%q%> Date displayed as the quarter of the year (1 to 4).
- <%**y**%> Number of the day of the year (1 to 366).
- <%yy%> Last two digits of the year (01 to 99).
- <%**yyyy**%> Full year (0100 to 9999).
- <%hy%> Date displayed as the half of the year (1 to 2).
- <%loc_gg%> Indicates period/era using the current user regional and language settings.
- <%loc_sdate%> Indicates the date in the short format using the current user regional and language settings.
- <%loc_ldate%> Indicates the date in the long format using the current user regional and language settings.
- <%loc_dsep%> Indicates the date separator using the current user regional and language settings (/).
- <%h%> Hour in one or two digits, as needed (0 to 23).
- <%**hh**%> Hour in two digits (00 to 23).
- <%**n**%> Minute in one or two digits, as needed (0 to 59).
- <%**nn**%> Minute in two digits (00 to 59).
- <%s%> Second in one or two digits, as needed (0 to 59).
- <%**ss**%> Second in two digits (00 to 59).
- <%AM/PM%> Twelve-hour clock with the uppercase letters "AM" or "PM", as appropriate. (Use the AMPM property to specify the name of the AM and PM indicators). You can use the <%loc_AM/PM%> that indicates the time marker such as AM or PM using the current user regional and language settings. You can use <%loc_A/P%> that indicates the one character time marker such as A or P using the current user regional and language settings
- <%loc_AM/PM%> Indicates the time marker such as AM or PM using the current user regional and language settings.
- <%loc_A/P%> Indicates the one character time marker such as A or P using the current user regional and language settings.
- <%loc_time%> Indicates the time using the current user regional and language settings.
- <%loc_time24%> Indicates the time in 24 hours format without a time marker using the current user regional and language settings.
- <%loc_tsep%> indicates the time separator using the current user regional and language settings (:).

The following tags are displayed based on the user's Regional and Language Options:

- <%loc_sdate%> Indicates the date in the short format using the current user settings.
- <%loc_ldate%> Indicates the date in the long format using the current user settings.
- <%loc_d1%> Indicates day of week as a one-letter abbreviation using the current user settings.
- <%loc_d2%> Indicates day of week as a two-letters abbreviation using the current user settings.

- <%loc_d3%> equivalent with <%loc_ddd%>
- <%loc_ddd%> Indicates day of week as a three-letters abbreviation using the current user settings.
- <%loc_dddd%> Indicates day of week as its full name using the current user settings.
- <%loc_m1%> Indicates month as a one-letter abbreviation using the current user settings.
- <%loc_m2%> Indicates month as a two-letters abbreviation using the current user settings.
- <%loc_m3%> equivalent with <%loc_mmm%>
- <%loc_mmm%> Indicates month as a three-letters abbreviation using the current user settings.
- <%loc_mmmm%> Indicates month as its full name using the current user settings.
- <%loc_gg%> Indicates period/era using the current user settings.
- <%loc_dsep%> Indicates the date separator using the current user settings.
- <%loc_time%> Indicates the time using the current user settings.
- <%loc_time24%> Indicates the time in 24 hours format without a time marker using the current user settings.
- <%loc_AM/PM%> Indicates the time marker such as AM or PM using the current user settings.
- <%loc_A/P%> Indicates the one character time marker such as A or P using the current user settings.
- <%loc_tsep%> Indicates the time separator using the current user settings.
- <%loc_y%> Represents the Year only by the last digit, using current regional settings.
- <%loc_yy%> Represents the Year only by the last two digits, using current regional settings. A leading zero is added for single-digit years.
- <%loc_yyyy%> Represents the Year by a full four or five digits, depending on the calendar used. Thai Buddhist and Korean calendars have five-digit years. The "yyyy" pattern shows five digits for these two calendars, and four digits for all other supported calendars. Calendars that have single-digit or two-digit years, such as for the Japanese Emperor era, are represented differently. A single-digit year is represented with a leading zero, for example, "03". A two-digit year is represented with two digits, for example, "13". No additional leading zeros are displayed.

The HeaderDayLongLabel property supports the following built-in HTML tags:

- ... displays the text in **bold**
- <i> ... </i> displays the text in *italics*
- <u> ... </u> <u>underlines</u> the text
- <s> ... </s> Strike-through text
- <a id;options> ... displays an <u>anchor</u> element that can be clicked. An anchor is a

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

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

- exp, stores the plain text to be shown once the user clicks the anchor, such as "<a ;exp=show lines>"
- e64, encodes in BASE64 the HTML text to be shown once the user clicks the anchor, such as "<a

;e64=gA8ABmABnABjABvABshIAOQAEAAHAAGESikWio+ABzABohp3iELABpABu " that displays show lines- in gray when the user clicks the + anchor. The "gA8ABmABnABjABvABshIAOQAEAAHAAGESikWio+ABzABohp3iELABpABuABljY string encodes the "<fgcolor 808080>show lines<a>-</fgcolor>" The Decode64Text/Encode64Text methods of the eXPrint can be used to decode/encode e64 fields.

Any ex-HTML caption can be transformed to an expandable-caption, by inserting the anchor ex-HTML tag. For instance, "<solidline>Header</solidline>

Line1<r><a ;exp=show lines>+
Line2
Line3" shows the Header in underlined and bold on the first line and Line1, Line2, Line3 on the rest. The "show lines" is shown instead of Line1, Line2, Line3 once the user clicks the + sign.

- ... displays portions of text with a different font and/or different size. For instance, the "bit" draws the bit text using the Tahoma font, on size 12 pt. If the name of the font is missing, and instead size is present, the current font is used with a different size. For instance, "bit" displays the bit text using the current font, but with a different size.
- **<fgcolor rrggbb>** ... **</fgcolor>** or **<**fgcolor=rrggbb> ... **</**fgcolor> displays text with a specified foreground color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- <bgcolor rrggbb> ... </bgcolor> or <bgcolor=rrggbb> ... </bgcolor> displays text with a specified background color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- <solidline rrggbb> ... </solidline> or <solidline=rrggbb> ... </solidline> draws a solidline 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
- number[:width] 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.
- key[:width] 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 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 FFFFF;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:



<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 Schedule.HeaderDayShortLabel as String

Specifies the short label for header days.

Туре	Description
String	A String expression that indicates the alternate labels + an expression that can be used to show a specific HTML label on certain conditions.

By default, the HeaderDayShortLabel property is "<%mmm%> <%d%><|><%m3%> <%d%><|><%d%><=>(((day(value) = 1) or (month(value+1) != month(value))) ? -1 : 2)". The HeaderDayShortLabel property defines the HTML label to show on date's header when no time scale is available (the schedule view has been shrink until the time scale is hidden). The <u>HeaderDayLongLabel</u> property defines the alternate HTML labels being shown on date's header, when the time scale is visible. The <u>HeaderDayHeight</u> property indicates the height of the day's header. The <u>HeaderDayLabel</u> property specifies the HTML date-format to be shown on the calendar's header.

The following screen shot shows the short header's schedule view if using "<sha> <%mmm%> <%d%><|><%m3%> <%d%><|><%d%><=>(((day(value) = 1) or (month(value+1) != month(value))) ? -1 : 2)"

29	30	July31	August1	2	3	4
			8/1/2012 - 8/31/2012			
5	6	7	8	9	10	11
8/1/2012 - 8/31/201	2	1			1	
		8/7/2012 - 8/9/2012	2			
12	13	14	15	16	17	18
8/1/2012 - 8/31/201	2					1
				8/16/2012 - 8/30/2	2012	
19	20	21	22	23	24	25
8/1/2012 - 8/31/201	2					
8/16/2012 - 8/30/20	12					
26	27	28	29	30	August31	September1
8/1/2012 - 8/31/201	2				1	
8/16/2012 - 8/30/20	12	di.		44 		

The following screen shot shows the short header's schedule view if using ""<%mmm%> <%d%><|><%m3%> <%d%><|><%d%><=>(((day(value) = 1) or (month(value+1) != month(value))) ? -1 : 2)" " (default)

29	30	July 31 🕻	August 1	2	3	4
		-	8/1/2012 - 8/31/20	12		
5	6	7	8	9	10	11
/1/2012 - 8/31/201	2	1	1		1	
		8/7/2012 - 8/9/2012	i i	1		
12	13	14	15	16	17	18
/1/2012 - 8/31/201	2	1		1		1
				8/16/2012 - 8/30/2	012	
19	20	21	22	23	24	25
/1/2012 - 8/31/201	2		1			1
/16/2012 - 8/30/20	12					
26	27	28	29	30	August 31	September 1
/1/2012 - 8/31/201	2		1		1	
16/2012 - 8/30/20	12	de la companya de la		1.12 Alter		

The first part of the HeaderDayShortLabel's format is similar with the <u>HeaderDayLongLabel</u>'s format, and it indicates a list of alternate labels separated by the <|>, while the last part of the format indicates an expression that determines whether a specified HTML label is being displayed on a certain condition.

In conclusion, the format of the HeaderDayShortLabel property is: "ALT1[<|>ALT2<|>...] [<=>EXPR]"

- ALT defines a HTML label that supports a lot of predefined built-in tags.
- EXPR defines an expression to specify which HTML label to be shown when a condition is met. The **EXPR** supports the **value** keyword that specifies the date being queried. The EXPR supports the <u>following</u> operators and functions.

For instance, HeaderDayLongLabel could be

- "<%d%>", always displays day's number.
- "<%m3%> <%d%>", always displays the three-letters month, followed by the day's number.
- "<%mmmm%> <%d%><|><%m3%> <%d%><|><%d%><=>(((day(value) = 1) or (month(value+1) != month(value))) ? -1 : 2)", displays the name of the month and the day number for the first and last days of the month, and in rest just the day number.

When the user resizes the schedule view, the control searches for the best fit, and the date header displays the proper HTML label, based on its width, so no ... (three dots) is shown in the labels as much as possible.

The HeaderDayLongLabel property supports the following built-in tags:

• <%d%> - Day of the month in one or two numeric digits, as needed (1 to 31).

- <%dd%> Day of the month in two numeric digits (01 to 31).
- <%d1%> First letter of the weekday (S to S). (Use the <u>WeekDays</u> property to specify the name of the days in the week)
- <%loc_d1%> Indicates day of week as a one-letter abbreviation using the current user settings.
- <%d2%> First two letters of the weekday (Su to Sa). (Use the <u>WeekDays</u> property to specify the name of the days in the week)
- <%loc_d2%> Indicates day of week as a two-letters abbreviation using the current user settings.
- <%d3%> First three letters of the weekday (Sun to Sat). (Use the <u>WeekDays</u> property to specify the name of the days in the week)
- <%loc_d3%> equivalent with <%loc_ddd%>
- <%ddd%> First three letters of the weekday (Sun to Sat). (Use the <u>WeekDays</u> property to specify the name of the days in the week). You can use the <%loc_ddd%> that indicates the day of week as a three-letter abbreviation using the current user regional and language settings.
- <%loc_ddd%> Indicates the day of week as a three-letter abbreviation using the current user regional and language settings.
- <%dddd%> Full name of the weekday (Sunday to Saturday). (Use the <u>WeekDays</u> property to specify the name of the days in the week). You can use the <%loc_dddd%> that indicates day of week as its full name using the current user regional and language settings.
- <%loc_dddd%> Indicates day of week as its full name using the current user regional and language settings.
- <%i%> Displays the number instead the date. For instance, you can display numbers as 1000, 1001, 1002, 1003, instead dates. (the valid range is from -647,434 to 2,958,465)
- <%**w**%> Day of the week (1 to 7).
- <%**ww**%> Week of the year (1 to 53).
- <%m%> Month of the year in one or two numeric digits, as needed (1 to 12).
- <%mr%> Month of the year in Roman numerals, as needed (I to XII).
- <%mm%> Month of the year in two numeric digits (01 to 12).
- <%m1%> First letter of the month (J to D). (Use the <u>MonthNames</u> property to specify the name of the months in the year)
- <%loc_m1%> Indicates month as a one-letter abbreviation using the current user settings.
- <%m2%> First two letters of the month (Ja to De). (Use the <u>MonthNames</u> property to specify the name of the months in the year)
- <%loc_m2%> Indicates month as a two-letters abbreviation using the current user settings.
- <%m3%> First three letters of the month (Jan to Dec). (Use the <u>MonthNames</u> property to specify the name of the months in the year)

- <%loc_m3%> equivalent with <%loc_mmm%>
- <%mmm%> First three letters of the month (Jan to Dec). (Use the <u>MonthNames</u> property to specify the name of the months in the year). You can use the <%loc_mmm%> that indicates month as a three-letter abbreviation using the current user regional and language settings.
- <%loc_mmm%> Indicates month as a three-letter abbreviation using the current user regional and language settings.
- <%mmm%> Full name of the month (January to December). (Use the MonthNames property to specify the name of the months in the year). You can use the <%loc_mmm%> that indicates month as its full name using the current user regional and language settings.
- <%loc_mmmm%> Indicates month as its full name using the current user regional and language settings.
- <%q%> Date displayed as the quarter of the year (1 to 4).
- $\langle \mathbf{y} \mathbf{y} \rangle$ Number of the day of the year (1 to 366).
- <%**yy**%> Last two digits of the year (01 to 99).
- <%yyyy%> Full year (0100 to 9999).
- <%hy%> Date displayed as the half of the year (1 to 2).
- <%loc_gg%> Indicates period/era using the current user regional and language settings.
- <%loc_sdate%> Indicates the date in the short format using the current user regional and language settings.
- <%loc_ldate%> Indicates the date in the long format using the current user regional and language settings.
- <%loc_dsep%> Indicates the date separator using the current user regional and language settings (/).
- <%h%> Hour in one or two digits, as needed (0 to 23).
- <%**hh**%> Hour in two digits (00 to 23).
- <%**n**%> Minute in one or two digits, as needed (0 to 59).
- <%nn%> Minute in two digits (00 to 59).
- <%s%> Second in one or two digits, as needed (0 to 59).
- <%ss%> Second in two digits (00 to 59).
- <%AM/PM%> Twelve-hour clock with the uppercase letters "AM" or "PM", as appropriate. (Use the <u>AMPM</u> property to specify the name of the AM and PM indicators). You can use the <%loc_AM/PM%> that indicates the time marker such as AM or PM using the current user regional and language settings. You can use <%loc_A/P%> that indicates the one character time marker such as A or P using the current user regional and language settings.
- <%loc_AM/PM%> Indicates the time marker such as AM or PM using the current user regional and language settings.
- <%loc_A/P%> Indicates the one character time marker such as A or P using the current user regional and language settings.

- <%loc_time%> Indicates the time using the current user regional and language settings.
- <%loc_time24%> Indicates the time in 24 hours format without a time marker using the current user regional and language settings.
- <%loc_tsep%> indicates the time separator using the current user regional and language settings (:)
- <%loc_y%> Represents the Year only by the last digit, using current regional settings.
- <%loc_yy%> Represents the Year only by the last two digits, using current regional settings. A leading zero is added for single-digit years.
- <%loc_yyyy%> Represents the Year by a full four or five digits, depending on the calendar used. Thai Buddhist and Korean calendars have five-digit years. The "yyyy" pattern shows five digits for these two calendars, and four digits for all other supported calendars. Calendars that have single-digit or two-digit years, such as for the Japanese Emperor era, are represented differently. A single-digit year is represented with a leading zero, for example, "03". A two-digit year is represented with two digits, for example, "13". No additional leading zeros are displayed

The following tags are displayed based on the user's Regional and Language Options:

- <%loc_sdate%> Indicates the date in the short format using the current user settings.
- <%loc_ldate%> Indicates the date in the long format using the current user settings.
- <%loc_d1%> Indicates day of week as a one-letter abbreviation using the current user settings.
- <%loc_d2%> Indicates day of week as a two-letters abbreviation using the current user settings.
- <%loc_d3%> equivalent with <%loc_ddd%>
- <%loc_ddd%> Indicates day of week as a three-letters abbreviation using the current user settings.
- <%loc_dddd%> Indicates day of week as its full name using the current user settings.
- <%loc_m1%> Indicates month as a one-letter abbreviation using the current user settings.
- <%loc_m2%> Indicates month as a two-letters abbreviation using the current user settings.
- <%loc_m3%> equivalent with <%loc_mmm%>
- <%loc_mmm%> Indicates month as a three-letters abbreviation using the current user settings.
- <%loc_mmmm%> Indicates month as its full name using the current user settings.
- <%loc_gg%> Indicates period/era using the current user settings.
- <%loc_dsep%> Indicates the date separator using the current user settings.

- <%loc_time%> Indicates the time using the current user settings.
- <%loc_time24%> Indicates the time in 24 hours format without a time marker using the current user settings.
- <%loc_AM/PM%> Indicates the time marker such as AM or PM using the current user settings.
- <%loc_A/P%> Indicates the one character time marker such as A or P using the current user settings.
- <%loc_tsep%> Indicates the time separator using the current user settings.
- <%loc_y%> Represents the Year only by the last digit, using current regional settings.
- <%loc_yy%> Represents the Year only by the last two digits, using current regional settings. A leading zero is added for single-digit years.
- <%loc_yyyy%> Represents the Year by a full four or five digits, depending on the calendar used. Thai Buddhist and Korean calendars have five-digit years. The "yyyy" pattern shows five digits for these two calendars, and four digits for all other supported calendars. Calendars that have single-digit or two-digit years, such as for the Japanese Emperor era, are represented differently. A single-digit year is represented with a leading zero, for example, "03". A two-digit year is represented with two digits, for example, "13". No additional leading zeros are displayed

The HeaderDayLongLabel property supports the following built-in HTML tags:

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

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

- exp, stores the plain text to be shown once the user clicks the anchor, such as "<a ;exp=show lines>"
- $\circ~$ e64, encodes in BASE64 the HTML text to be shown once the user clicks the anchor, such as "<a $\!$

;e64=gA8ABmABnABjABvABshIAOQAEAAHAAGESikWio+ABzABohp3iELABpABu

" that displays show lines- in gray when the user clicks the + anchor. The "gA8ABmABnABjABvABshIAOQAEAAHAAGESikWio+ABzABohp3iELABpABuABljY string encodes the "<fgcolor 808080>show lines<a>-</fgcolor>" The Decode64Text/Encode64Text methods of the eXPrint can be used to decode/encode e64 fields.

Any ex-HTML caption can be transformed to an expandable-caption, by inserting the anchor ex-HTML tag. For instance, "<solidline>Header</solidline>

Line1<r><a ;exp=show lines>+
Line2
Line3" shows the Header in underlined and bold on the first line and Line1, Line2, Line3 on the rest. The "show lines" is shown instead of Line1, Line2, Line3 once the user clicks the + sign.

- ... displays portions of text with a different font and/or different size. For instance, the "bit" draws the bit text using the Tahoma font, on size 12 pt. If the name of the font is missing, and instead size is present, the current font is used with a different size. For instance, "bit" displays the bit text using the current font, but with a different size.
- <fgcolor rrggbb> ... </fgcolor> or <fgcolor=rrggbb> ... </fgcolor> displays text with
 a specified foreground color. The rr/gg/bb represents the red/green/blue values of the
 color in hexa values.
- <bgcolor rrggbb> ... </bgcolor> or <bgcolor=rrggbb> ... </bgcolor> displays text with a specified background color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- <solidline rrggbb> ... </solidline> or <solidline=rrggbb> ... </solidline> draws a solidline 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
- number[:width] 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.

- key[:width] inserts a custom size picture into the text being previously loaded using the HTMLPicture property. The Key parameter indicates the key of the picture being displayed. The Width parameter indicates a custom size, if you require to stretch the picture, else the original size of the picture is used.
- & glyph characters as & amp; (&), &It; (<), > (>), &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 FFFFF;1;1>gradient-center</gra>" generates the following picture:

gradient-center

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

outlined

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

following picture:

shadow

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

outline anti-aliasing

The EXPR supports the following operators and methods:

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#, #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
- Itrim (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)

property Schedule.HeaderGroupHeight as Double

Indicates the height of the group's header.

Туре	Description
Double	A double expression that indicates the height of the header to be shown for groups, in the schedule view. A positive value is multiplied with the font's height, while a negative value indicates a fixed height.

By default, the HeaderGroupHeight property is 1, which indicates that the height of the group's header is the same as the font's height. The HeaderGroupHeight property determines the height of the header by multiplying the value with the height of the current font, if the value is positive, else if the value is negative it indicates a fixed height. The Font property indicates the control's font. Use the HeaderGroupHeight property to programmatically extend the header's height. The HeaderGroupHeight property on 0, indicates that no header is shown for groups. The Caption/Title property of the Group is displayed on the group's header.

The <u>HeaderDayHeight</u> property indicates the height of the header that displays dates in the schedule view.

property Schedule.HighlightDate(Date as Date) as Variant

Highlights the specified date.

Туре	Description
Date as Date	A DATE expression that specifies the date to be highlighted
Variant	A long expression that specifies the color to be applied (RGB color), a string expression that specifies the list of colors to be applied, a uni-dimensional safe array of long expression that specifies the colors to be applied.

By default, the HighlightDate property returns an empty value, which indicates that no color is applied to the date. The highlight can be applied to the date shown in the calendar panel, or in the date header being shown on the schedule panel. Use the <u>ShowHighlightDate</u> property to specify whether the highlighted dates are shown on calendar or/and schedule panels. The <u>HighlightDate</u> property highlights/un-highlights the specified date with the giving color(s).

The following VB6 samples shows how you can set the HighlightDate property:

- Schedule1.HighlightDate(#11/12/2013#) = 255
- Schedule1.HighlightDate(#11/12/2013#) = RGB(255, 0, 0)
- Schedule1.HighlightDate(#11/12/2013#) = "255,65535"
- Schedule1.HighlightDate(#11/12/2013#) = Array(RGB(255, 0, 0),RGB(255,255,0))

The <u>HighlightEvent</u> property highlights dates in the calendar panel, when it contains events. The <u>ScheduleHighlightEvent</u> property specifies the visual appearance of dates with events in the schedule panel.

property Schedule.HTMLPicture(Key as String) as Variant

Adds or replaces a picture in HTML captions.

Туре	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.
	The HTMLPicture specifies the picture being associated to a key. It can be one of the followings:
Variant	 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 <u>eximages</u> tool to save your picture as base64 encoded format. A Picture object that indicates the picture being added or replaced. (A Picture object implements IPicture interface),
	If empty, the picture being associated to a key is removed. If the key already exists the new picture is replaced. If the key is not empty, and it doesn't not exist a new picture is added

The HTMLPicture property handles a collection of custom size picture being displayed in the HTML captions, using the 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 "pic1" sequence in HTML captions, displays the pic1 picture. On return, the HTMLPicture property retrieves a Picture object (this implements the IPictureDisp interface)

The <u>Images</u> method loads icons to the control, HTMLPicture assigns a key to a picture object, and the <u>Pictures</u> collection handles the identifiers of the pictures that can be used in the Pictures or <u>ExtraPictures</u> properties.

In order to display an icon or a picture in the control you need first to load the icons or the pictures you plan to display, using the <u>Images</u> method, <u>HTMLPicture</u>, or <u>Add</u> method of the ExPictures collection. The Images collection can display only 16x16 icons, while the <u>HTMLPicture</u>, or <u>Add</u> method can load and display custom sized pictures. The <u>Width/Height</u> property specifies the width and height of the picture to be displayed in the event's body.

The event can display icons, pictures several ways as follows:

- Using the HTML tag, if used in any label properties such as LongLabel or ExtraLabel property. The ShortLabel property can NOT display images or HTML font attributes. For instance, this option can be used to display a default icon or picture for all events in the control using the DefaultEventLongLabel property. Also, the TAG can be used to any label or caption property that supports HTML format.
- Using the <u>Pictures</u> property of the Event object, which indicates a list of pictures from Pictures collection to be displayed on the event's body. The <u>PicturesAlign</u> property indicates the alignment of the pictures relative to the borders of the event. The hand cursor is shown, if the ExPicture.<u>ShowCursorHand</u> property is set, when the cursor hovers the picture. The <u>PictureClick</u> event is fired if the user clicks a picture on event's picture. You can use the <u>PictureFromPoint</u> property to get the identifier of the picture from the cursor.
- Using the <u>ExtraPictures</u> property of the Event object, which indicates a list of pictures from Pictures collection to be displayed on the event's body. The <u>ExtraPicturesAlign</u> property indicates the alignment of the pictures relative to the borders of the event. The hand cursor is shown, if the ExPicture.<u>ShowCursorHand</u> property is set, when the cursor hovers the picture. The <u>PictureClick</u> event is fired if the user clicks a picture on event's picture. You can use the <u>PictureFromPoint</u> property to get the identifier of the picture from the cursor.

The <u>Pictures</u> and <u>ExtraPictures</u> may display one or more pictures at the time. The , character indicates the separator of pictures in the same line, while the / character divides the lines to show the pictures. For instance, "1,2" displays icon with the index 1 and 2 on the same line, while the "1/2,pic1" displays the first icon on the first line, the second icon and the picture pic1 on the second line.

property Schedule.hWnd as Long

Retrieves the control's window handle.

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

Use the hWnd property to get the control's main window handle. Use the <u>Calendar.hWnd</u> property to get the handle of the calendar panel. 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. Use the <u>Parent</u> property to move the calendar panel to other place.

method Schedule.Images (Handle as Variant)

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

Туре	Description
	The Handle parameter can be:
Handle as Variant	 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 (<i>string, loads the</i> <i>icon using its path</i>) 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" (<i>string, loads icons using base64</i> <i>encoded string</i>) A reference to a Microsoft ImageList control (mscomctl.ocx, MSComctlLib.ImageList type) that holds the icons to add (<i>object, loads icons from a</i> <i>Microsoft ImageList control</i>) 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 (<i>object, loads icon from a Picture object</i>) 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 IIVal field, as VT_18 type. The LONGLONG / LONG_PTR isint64, 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 holder. The <u>ImageSize</u> property defines the size (width/height) of the icons within the control's Images collection. The <u>ShowImageList</u> property available for the /COM shows or hides the control's images holder at design mode. Use the <u>ReplaceIcon</u> method to add, remove or clear icons in the control's images collection.

The Images method loads icons to the control, <u>HTMLPicture</u> assigns a key to a picture object, and the <u>Pictures</u> collection handles the identifiers of the pictures that can be used in the Pictures or <u>ExtraPictures</u> properties.

In order to display an icon or a picture in the control you need first to load the icons or the pictures you plan to display, using the <u>Images</u> method, <u>HTMLPicture</u>, or <u>Add</u> method of the ExPictures collection. The Images collection can display only 16x16 icons, while the <u>HTMLPicture</u>, or <u>Add</u> method can load and display custom sized pictures. The <u>Width/Height</u> property specifies the width and height of the picture to be displayed in the event's body.

The event can display icons, pictures several ways as follows:

- Using the HTML tag, if used in any label properties such as LongLabel or ExtraLabel property. The ShortLabel property can NOT display images or HTML font attributes. For instance, this option can be used to display a default icon or picture for all events in the control using the DefaultEventLongLabel property. Also, the TAG can be used to any label or caption property that supports HTML format.
- Using the <u>Pictures</u> property of the Event object, which indicates a list of pictures from Pictures collection to be displayed on the event's body. The <u>PicturesAlign</u> property indicates the alignment of the pictures relative to the borders of the event. The hand cursor is shown, if the ExPicture.<u>ShowCursorHand</u> property is set, when the cursor hovers the picture. The <u>PictureClick</u> event is fired if the user clicks a picture on event's picture. You can use the <u>PictureFromPoint</u> property to get the identifier of the picture from the cursor.
- Using the <u>ExtraPictures</u> property of the Event object, which indicates a list of pictures from Pictures collection to be displayed on the event's body. The <u>ExtraPicturesAlign</u> property indicates the alignment of the pictures relative to the borders of the event. The hand cursor is shown, if the ExPicture.<u>ShowCursorHand</u> property is set, when the cursor hovers the picture. The <u>PictureClick</u> event is fired if the user clicks a picture on event's picture. You can use the <u>PictureFromPoint</u> property to get the identifier of the picture from the cursor.

The <u>Pictures</u> and <u>ExtraPictures</u> may display one or more pictures at the time. The , character indicates the separator of pictures in the same line, while the *I* character divides the lines to show the pictures. For instance, "1,2" displays icon with the index 1 and 2 on the same line, while the "1/2,pic1" displays the first icon on the first line, the second icon and the picture pic1 on the second line.

property Schedule.ImageSize as Long

l

Retrieves or sets the size of icons the control displays..

Туре	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 Schedule.Layout as String

Saves or loads the control's layout, such as positions of the columns, scroll position, filtering values.

Туре	Description
String	A String expression that specifies the control's layout.

You can use the Layout property to store the control's layout and to restore the layout later. For instance, you can save the control's Layout property to a file when the application is closing, and you can restore the control's layout when the application is loaded. The Layout property saves almost all of the control's properties that user can change at runtime (like changing the event's position by drag and drop). The Layout property does NOT save the control's data, so the Layout property should be called once you loaded the data from your database, xml or any other alternative. Once the data is loaded, you can call the Layout property to restore the View as it was saved. Before closing the application, you can call the Layout property and save the content to a file for reading next time the application is opened.

The Layout property saves/loads the following information:

- <u>OnResizeControl</u> property, that specifies how the panels are arranged, sized and so on.
- The selection in the calendar panel (selected date)
- The date begin browsed in the calendar panel, <u>Date</u> property
- The alignment, position and width for each time scale
- The alignment, position and width for each time group
- The selection in the schedule panel (selected events)
- The schedule view scrolling position

These properties are serialized as a string.

method Schedule.LoadXML (Source as Variant)

Loads an XML document from the specified location, using MSXML parser.

Туре	Description
Source as Variant	An indicator of the object that specifies the source for the XML document. The object can represent a file name, a URL, an IStream, a SAFEARRAY, or an IXMLDOMDocument.
Return	Description
Boolean	A boolean expression that specifies whether the XML document is loaded without errors. If an error occurs, the method retrieves a description of the error occurred.

The LoadXML method uses the MSXML (MSXML.DOMDocument, XML DOM Document) parser to load XML documents, previously saved using the <u>SaveXML</u> method. Before calling the LoadXML you may call the <u>ClearAll</u> method to empty the control's content. The LoadXML method fires <u>AddEvent</u> event for each loaded event. It is good to know, that the LoadXML method does not clear the Events collection, and this is happen to allow you load multiple XML files, so it can add new events each time.

The <u>XML format</u> looks like follows:

 - <Content Author Component Version> <DateFormat Separator Short Long /> <TimeFormat Separator Format AM PM /> <NumberFormat Decimal /> -<Calendar WeekDays MonthNames AMPM FirstWeekDay NonworkingDays ShortDateFormat LongDateFormat ShortTimeFormat LongTimeFormat GroupHighlightEven ShowHighlightEvent DisableZoneFormat MinDate MaxDate ShowNonMonthDays Date SingleSel HideSel ... > - < Selection > <Date Value .../> </Selection> <HighlightEvent Bold Italic Underline StrikeOut FontSize .../> </Calendar> - <TimeScales DayStartTime DayEndTime> <TimeScale TimeZone AlignLeft MinWidth MaxWidth Width AllowResize MajorRuler MajorLabel MajorPlainText MinorRuler MinorLabel MinorPlainText Visible Caption CaptionAlign ToolTip UserData MinorLabelColor .../> </TimeScales> - <Groups ApplyGroupingColors DisplayGroupingButton GroupHighlightEvent ShowGroupingEvents> -<Group ID Visible Caption Title ToolTip Alignment UserData> <CalendarHighlightEvent Bold Italic Underline StrikeOut FontSize ... /> <ScheduleHighlightEvent Bold Italic Underline StrikeOut FontSize ... /> </Group> </Groups> - <NonworkingPatterns> <NonworkingPattern ID PatternType PatternColor PatternFrameColor .../> </NonworkingPatterns> -

<NonworkingTimes ShowNonworkingTime> <NonworkingTime Expression StartTime EndTime IDNonworkingPattern .../> </NonworkingTimes> -<MarkZones> <MarkZone Key Start End ShortLabel LongLabel GroupID PatternType PatternColor PatternFrameColor ... /> </MarkZones> -<MarkTimes ShowMarkTime> <MarkTime Key Time Line TimeScaleLine LineColor TimeScaleLineColor Label TimeScaleLabel LabelAlign TimeScaleLabelAlign Movable StatusEventBackColor UserData ... /> </MarkTimes> - <Pictures ShowEventPictures> <Picture Key Content Width Height ShowHandCursor Enabled ... /> </Pictures> <Icons Content/> -<Events HeaderDayLongLabel HeaderDayShortLabel CreateEventLabel CreateEventLabelAlign UpdateEventsLabel UpdateEventsLabelAlign DefaultEventLongLabel DefaultEventTooltip DefaultEventPaddingLeft DefaultEventPaddingTop DefaultEventPaddingRight DefaultEventPaddingBottom ...> < Event Start End GroupID Selectable ShowStatus ShortLabel LongLabel LabelAlign ExtraLabelAlign Pictures ... /> </Events> </Content>

property Schedule.MarkTimeFromPoint (X as OLE_XPOS_PIXELS, Y as OLE_YPOS_PIXELS) as MarkTime

Gets the MarkTime object from the cursor, in the schedule panel.

Туре	Description
X as OLE_XPOS_PIXELS	A single that specifies the current X location of the mouse pointer. The x values is always expressed in client coordinates.
Y as OLE_YPOS_PIXELS	A single that specifies the current Y location of the mouse pointer. The y values is always expressed in client coordinates.
<u>MarkTime</u>	A MarkTime object, or a timer from the cursor.

The <u>MouseMove</u> event is generated continually as the mouse pointer moves across objects. During the MouseMove event you can call the <u>ShowToolTip</u> method to display any custom tooltip. During the <u>Click</u> or <u>RClick</u> event you can get an UI part of the control using one of the following properties. *All ...FromPoint properties can be use such as ...FromPoint(-1,-1) to get the UI part of the control from the current mouse position, in other words, you do not have to pass any X, Y coordinates.*

You can get UI parts from the cursor, using any of the following ... FromPoint properties:

- Calendar. <u>DateFromPoint</u> property gets the date from the cursor in the Calendar panel.
- AnchorFromPoint property gets the anchor (<a>) element from the cursor.
- <u>DateTimeFromPoint</u> property gets the date/time from the cursor, in the Schedule panel.
- <u>TimeFromPoint</u> property gets the time from the cursor, in the Schedule panel.
- EventFromPoint property gets the Event object from the cursor.
- <u>GroupFromPoint</u> property retrieves the <u>Group</u> object from the cursor. Use the GroupHeaderFromPoint property to the Group object when cursor hovers the group's header.
- <u>GroupHeaderFromPoint</u> property retrieves the <u>Group</u> object from the cursor, when the cursor hovers just the group's header.
- <u>MarkTimeFromPoint</u> property gets the <u>MarkTime</u> object from the cursor. A MarkTime object is identified by a date/time where it is displaying.
- <u>MarkZoneFromPoint</u> property gets the <u>MarkZone</u> object from the cursor. A MarkZone object is identified by a starting date/time and ending date/time where it is displaying.
- <u>NonworkingTimeFromPoint</u> property indicates the <u>NonworkingTime</u> object from the cursor.
- <u>PictureFromPoint</u> property indicates the key of the picture from the cursor.
- <u>TimeScaleFromPoint</u> property returns the <u>TimeScale</u> object from the cursor.

All ... FromPoint properties that returns an Object, may return Nothing, Empty or NULL, if no

object is found.

property Schedule.MarkTimes as MarkTimes

Gets the MarkTimes collection of the scheduler.

Туре	Description
<u>MarkTimes</u>	Returns a reference to the schedule's MarkTimes collection, which holds a set of MarkTime objects.

The MarkTimes property exports a collection of MarkTime objects, also called timers. The <u>MarkTime</u> object indicates a line in the schedule view, at a specified time. The <u>Add</u> method of <u>MarkTimes</u> collection adds a new timer to the schedule view. The MarkTimes collection is accessible through the MarkTimes property of the control. The <u>ShowMarkTime</u> property indicates whether the schedule view displays timers. The <u>AllowMoveMarkTime</u> property indicates the keys to allow user to move timers (with the <u>Movable</u> property on True). The <u>MarkTimeFromPoint</u> property indicates the timer from the cursor.

The MarkTime object, also called **timer**, can be used to:

- show a line of different styles on the schedule view, at specified time
- show a HTML label at specified time
- highlights the events that intersect with the timer

The following screen shot shows the control's timers (red and black/blue arrow):

Tuesday, May 15, 2012	Wednesday, May 16, 2012	-
5:30 AM - 11 10 AM 5/15/2 10:40 AM - 12:30 PM	AM 2012 11:10 AM - 3:20 PM 11:20 AM - 12:10 PM	08:00 AM :20 :40 09:00 AM :20 :40 10:00 AM :20 10:16 AM :40 11:00 AM :20 :40
1:20 PM - 3:20 PM	12:10 PM - 3:00 PM	12:00 PM :20 :40 01:00 PM :20 :40 :40 :20 :40 :20 :40 :20 :40 :20 :40 :40 :20 :20 :40 :20 :40 :20 :20 :40 :20 :20 :40 :20 :20 :20 :20 :20 :20 :20 :2
		02:00 PM :20 :40 03:00 PM :20 :40 ····································

property Schedule.MarkZoneFromPoint (X as OLE_XPOS_PIXELS, Y as OLE_YPOS_PIXELS) as MarkZone

Retrieves the MarkZone from the cursor, in the schedule panel.

Туре	Description
X as OLE_XPOS_PIXELS	A single that specifies the current X location of the mouse pointer. The x values is always expressed in client coordinates.
Y as OLE_YPOS_PIXELS	A single that specifies the current Y location of the mouse pointer. The y values is always expressed in client coordinates.
<u>MarkZone</u>	A MarkZone object, or a time-zone from the cursor.

The <u>MouseMove</u> event is generated continually as the mouse pointer moves across objects. During the MouseMove event you can call the <u>ShowToolTip</u> method to display any custom tooltip. During the <u>Click</u> or <u>RClick</u> event you can get an UI part of the control using one of the following properties. *All ...FromPoint properties can be use such as ...FromPoint(-1,-1) to get the UI part of the control from the current mouse position, in other words, you do not have to pass any X, Y coordinates.*

You can get UI parts from the cursor, using any of the following ... FromPoint properties:

- Calendar. <u>DateFromPoint</u> property gets the date from the cursor in the Calendar panel.
- AnchorFromPoint property gets the anchor (<a>) element from the cursor.
- <u>DateTimeFromPoint</u> property gets the date/time from the cursor, in the Schedule panel.
- <u>TimeFromPoint</u> property gets the time from the cursor, in the Schedule panel.
- EventFromPoint property gets the Event object from the cursor.
- <u>GroupFromPoint</u> property retrieves the <u>Group</u> object from the cursor. Use the GroupHeaderFromPoint property to the Group object when cursor hovers the group's header.
- <u>GroupHeaderFromPoint</u> property retrieves the <u>Group</u> object from the cursor, when the cursor hovers just the group's header.
- <u>MarkTimeFromPoint</u> property gets the <u>MarkTime</u> object from the cursor. A MarkTime object is identified by a date/time where it is displaying.
- <u>MarkZoneFromPoint</u> property gets the <u>MarkZone</u> object from the cursor. A MarkZone object is identified by a starting date/time and ending date/time where it is displaying.
- <u>NonworkingTimeFromPoint</u> property indicates the <u>NonworkingTime</u> object from the cursor.
- <u>PictureFromPoint</u> property indicates the key of the picture from the cursor.
- <u>TimeScaleFromPoint</u> property returns the <u>TimeScale</u> object from the cursor.

All ... FromPoint properties that returns an Object, may return Nothing, Empty or NULL, if no
object is found.

property Schedule.MarkZones as MarkZones

Retrieves the MarkZones collection of the scheduler.

Туре	Description
MarkZones	A MarkZones object that holds a collection of the MarkZone objects.

The MarkZones property gets the <u>MarkZones</u> collection. The MarkZones collection holds a set of <u>MarkZone</u> objects (also called time-zone). A MarkZone object holds information about a time-zone. A time-zone is identified by a <u>Start/End</u> date time, what can be highlighted in the schedule view. The <u>ShowMarkZone</u> property shows or hides the added time-zones. Use the <u>Add</u> method of the MarkZones collection to add a new time-zone to the control. The <u>MarkZoneFromPoint</u> property indicates the time-zone from the cursor.

The MarkZone object can:

- show a HTML/Image caption on a specified time-zone
- highlight a time-zone with a different background, pattern and so on, to indicate a restricted zone for instance.

A time-zone (<u>MarkZone</u> object) requires the <u>Start/End</u> to define the zone, while a timer (<u>MarkTime</u> object) requires a <u>Time</u>, that indicates where the timer is shown.

The following screen shot shows a time-zone in the control:



property Schedule.NonworkingPatterns as NonworkingPatterns

Retrieves the NonworkingPatterns collection of the scheduler.

NonworkingPatterns A NonworkingPatterns object that defines the pattern to be used by non-working time-intervals.	Туре	Description
	NonworkingPatterns	A NonworkingPatterns object that defines the pattern to be used by non-working time-intervals.

The <u>NonworkingPatterns</u> collection is accessible through the NonworkingPatterns property of the control. The <u>NonworkingPatterns</u> collection defines the patterns/colors to be used by non-working time intervals. In other words, each <u>NonworkingTime</u> object can associate a <u>NonworkingPattern</u> object that specifies the colors and the pattern to show the non-working zone. The <u>ShowNonworkingTime</u> property shows or hides the defined non-working intervals. The <u>NonworkingTimeFromPoint</u> property gets the non-working object from the cursor.

The <u>Add</u> method of the <u>NonworkingPatterns</u> collection defines a pattern/color. The <u>ID</u> property defines the identifier of the pattern/color. The <u>IDNonworkingPattern</u> property indicates the identifier of the NonworkingPattern object to be displayed on the non-working interval.

The <u>NonworkingPatterns</u> collection holds a list of <u>NonworkingPattern</u> objects. The <u>NonworkingDays</u> property of the calendar defines the days to be non-working in the calendar. The <u>NonworkingDaysPattern</u> and the <u>NonworkingDaysColor</u> which defines the pattern and the color to show the non-working days. The <u>FirstWeekDay</u> property indicates the first day of the week.

The <u>Add</u> method of the NonworkingTimes objects adds a new non-working time interval.

property Schedule.NonworkingTimeFromPoint (X as OLE_XPOS_PIXELS, Y as OLE_YPOS_PIXELS) as NonworkingTime

Retrieves the NonworkingTime from the cursor, in the schedule panel.

Туре	Description
X as OLE_XPOS_PIXELS	A single that specifies the current X location of the mouse pointer. The x values is always expressed in client coordinates.
Y as OLE_YPOS_PIXELS	A single that specifies the current Y location of the mouse pointer. The y values is always expressed in client coordinates.
Nonworking Time	A NonworkingTime object from the cursor.

The <u>MouseMove</u> event is generated continually as the mouse pointer moves across objects. During the MouseMove event you can call the <u>ShowToolTip</u> method to display any custom tooltip. During the <u>Click</u> or <u>RClick</u> event you can get an UI part of the control using one of the following properties. *All ...FromPoint properties can be use such as ...FromPoint(-1,-1) to get the UI part of the control from the current mouse position, in other words, you do not have to pass any X, Y coordinates.*

You can get UI parts from the cursor, using any of the following ... FromPoint properties:

- Calendar. <u>DateFromPoint</u> property gets the date from the cursor in the Calendar panel.
- AnchorFromPoint property gets the anchor (<a>) element from the cursor.
- <u>DateTimeFromPoint</u> property gets the date/time from the cursor, in the Schedule panel.
- <u>TimeFromPoint</u> property gets the time from the cursor, in the Schedule panel.
- EventFromPoint property gets the Event object from the cursor.
- <u>GroupFromPoint</u> property retrieves the <u>Group</u> object from the cursor. Use the GroupHeaderFromPoint property to the Group object when cursor hovers the group's header.
- <u>GroupHeaderFromPoint</u> property retrieves the <u>Group</u> object from the cursor, when the cursor hovers just the group's header.
- <u>MarkTimeFromPoint</u> property gets the <u>MarkTime</u> object from the cursor. A MarkTime object is identified by a date/time where it is displaying.
- <u>MarkZoneFromPoint</u> property gets the <u>MarkZone</u> object from the cursor. A MarkZone object is identified by a starting date/time and ending date/time where it is displaying.
- <u>NonworkingTimeFromPoint</u> property indicates the <u>NonworkingTime</u> object from the cursor.
- <u>PictureFromPoint</u> property indicates the key of the picture from the cursor.
- <u>TimeScaleFromPoint</u> property returns the <u>TimeScale</u> object from the cursor.

All ... FromPoint properties that returns an Object, may return Nothing, Empty or NULL, if no

object is found.

property Schedule.NonworkingTimes as NonworkingTimes

Retrieves the NonworkingTimes collection of the scheduler, to specify different non-working time for different days.

Туре	Description
<u>NonworkingTimes</u>	A NonworkingTimes object that holds a collection <u>NonworkingTime</u> objects.

The <u>NonworkingTimes</u> collection is accessible through the NonworkingTimes property of the control. The <u>Add</u> method of the NonworkingTimes objects adds a new non-working time interval. The <u>Expression</u> property indicates the expression that defines the dates to include the specified non-working interval. The <u>IsValid</u> property indicates whether the non-working expression is valid, and so, if it is visible or hidden. The <u>StartTime/EndTime</u> property defines the time to start/end the non-working time-zone. The <u>ShowNonworkingTime</u> property shows or hides the defined non-working intervals. The <u>NonworkingTimeFromPoint</u> property gets the non-working object from the cursor.

The NonworkingTimes object holds a collection of <u>NonworkingTime</u> objects. The NonworkingTime object indicates a time interval to be shown as non-working. Each NonworkingTime object can associate a <u>NonworkingPattern</u> object that specifies the colors and the pattern to show the non-working zone. The <u>NonworkingPatterns</u> collection is accessible through the <u>NonworkingPatterns</u> property of the control. The <u>NonworkingDays</u> property of the calendar defines the days to be non-working in the calendar.

The NonworkingTime's advantages are:

- highlight the interval of time as non-working with a different patterns, colors.
- any/all day can display different intervals of time as non-working
- you can specify the non-working interval using an expression, that defines the days where the non-working interval is shown.

The following screen shot shows different days with different non-working area:

	Monday, May 14	Tuesday, May 15	Wednesday, May 16	Thursday, May 17	Friday, May 18	Saturday, May 19	Sunday, May 20
07:00 AM :15 :30				7:30 AM -			
:45			7:40 AM -	12:00 PM			
US:00 AM			5.40 AM				
:30			- 10 H				
45: 09:00 AM		0.00 414 41.55 414	8:40 AM -		8:40 AM - 11:35 AM		
:15		9:00 AM - 11:00 AM	10.00 / 10.				
:30					8		
10:00 AM		///·		9:50 AM -			
:15			10:20 AM	12:55 PM			
:30			- 2:45 PM				
11:00 AM		·			ing in the second s		
:15							
:45					(and the second		
12:00 PM	11:55 AM - 2:25 PM		11:50 AM -		11:55 AM - 1:05 PM		
:15		11 12		10.00 044			
:45		12:40 PM - 4:05 PM		- 3:55 PM			
01:00 PM							
:30							
:45							
02:00 PM	This is an event						
:30							
:45 02:00 PM					H		
:15							
:30							
:45 04:00 PM							
:15			8				
:30							

method Schedule.OLEDrag ()

Causes a component to initiate an OLE drag/drop operation.

Туре

Description

property Schedule.OLEDropMode as exOLEDropModeEnum

Returns or sets how a target component handles drop operations

Туре	Description
<u>exOLEDropModeEnum</u>	An exOLEDropModeEnum expression that indicates the OLE Drag and Drop mode.

In the /NET Assembly, you have to use the AllowDrop property as explained here:

https://www.exontrol.com/sg.jsp?content=support/faq/net/#dragdrop

By default, the OLEDropMode property is exOLEDropNone. Currently, the ExSchedule control supports only manual OLE Drag and Drop operation. Use the <u>Background</u>(exDragDropBefore) property to specify the visual appearance for the dragging items, before painting the items. Use the <u>Background</u>(exDragDropAfter) property to specify the visual appearance for the dragging items, after painting the items. Use the <u>Background</u>(exDragDropList) property to specify the graphic feedback for the item from the cursor, while the OLE drag and drop operation is running. See the <u>OLEStartDrag</u> and <u>OLEDragDrop</u> events for more details about implementing drag and drop operations into the ExSchedule control.

property Schedule.OnResizeControl as OnResizeControlEnum

Specifies which panel is resized when the control is resized.

Туре

Description

OnResizeControlEnum An OnResizeControlEnum expression that specifies the operation the control does when it is resized.

By default, the OnResizeControl property is exResizePanelRight + exCalendarFit. This indicates that the right panel (such as schedule view) is resized when the control is resized, and the calendar panel always fit its portion (no partially view is allowed). The <u>ScrollBars</u> property indicates whether the control display the vertical or horizontal scroll bars. The AllowMoveSchedule property indicates the keys combination so the user can change the schedule view by dragging. The <u>AllowMoveSchedule</u> property on exDisallow, indicates that the user can not move the currently schedule date to a new position. The <u>AllowExchangePanels</u> property specifies whether the user can drag and drop a panel to a new position. The <u>FitSelToView</u> method restores the view to fit the selected dates. The control fires the <u>LayoutStartChanging</u>(exLayoutExchangePanels)/LayoutEndChanging(exLayoutExchangePanels) events once the user drags the panels from a side to another. The OnResizeControl property has effect only if the <u>Calendar.Parent</u> property is zero (default).

Also, you can use the OnResizeControl property to specify one of the followings:

- auto hide the calendar panel. Ability to hide the calendar section while the cursor is not in it (OnResizeControlEnum.exResizePanelRight Or OnResizeControlEnum.exCalendarFit Or OnResizeControlEnum.exCalendarAutoHide).
- hide completely the calendar section (exHideSplitter)
- specify the **alignment** of the calendar, as on the left or right side of the schedule view (OnResizeControlEnum.exChangePanels Or OnResizeControlEnum.exCalendarFit)
- full or **partially** view of the calendar panel (exResizePanelRight)
- disabling the control's vertical split bar (so user can not resize the fixed panel) (OnResizeControlEnum.exResizePanelRight Or OnResizeControlEnum.exDisableSplitter Or OnResizeControlEnum.exCalendarFit)
- use the Calendar.Parent property to move the calendar panel outside of the scheduler

The following samples hide the Calendar panel of the component:

VBA (MS Access, Excell...)

```
With Schedule1
.OnResizeControl = 768
.ScrollBars = 0
End With
```

VB6

With Schedule1 .**OnResizeControl** = OnResizeControlEnum.exHideSplitter Or OnResizeControlEnum.exChangePanels .ScrollBars = exNoScroll End With

VB.NET

With Exschedule1 .OnResizeControl = exontrol.EXSCHEDULELib.OnResizeControlEnum.exHideSplitter Or exontrol.EXSCHEDULELib.OnResizeControlEnum.exChangePanels .ScrollBars = exontrol.EXSCHEDULELib.ScrollBarsEnum.exNoScroll End With

VB.NET for /COM

```
With AxSchedule1

.OnResizeControl = EXSCHEDULELib.OnResizeControlEnum.exHideSplitter Or

EXSCHEDULELib.OnResizeControlEnum.exChangePanels

.ScrollBars = EXSCHEDULELib.ScrollBarsEnum.exNoScroll

End With
```

```
C++
```

Copy and paste the following directives to your header file as it defines the namespace 'EXSCHEDULELib' for the library: 'ExSchedule 1.0 Control Library'

```
#import <ExSchedule.dll>
using namespace EXSCHEDULELib;
```

```
*/
EXSCHEDULELib::ISchedulePtr spSchedule1 = GetDlgItem(IDC_SCHEDULE1)-
>GetControlUnknown();
spSchedule1-
>PutOnResizeControl(EXSCHEDULELib::OnResizeControlEnum(EXSCHEDULELib::exHid
| EXSCHEDULELib::exChangePanels));
spSchedule1->PutScrollBars(EXSCHEDULELib::exNoScroll);
```

C++ Builder

Schedule1->**OnResizeControl** = Exschedulelib_tlb::OnResizeControlEnum::exHideSplitter | Exschedulelib_tlb::OnResizeControlEnum::exChangePanels; Schedule1->ScrollBars = Exschedulelib_tlb::ScrollBarsEnum::exNoScroll;

C#

```
exschedule1.OnResizeControl =
exontrol.EXSCHEDULELib.OnResizeControlEnum.exHideSplitter |
exontrol.EXSCHEDULELib.OnResizeControlEnum.exChangePanels;
exschedule1.ScrollBars = exontrol.EXSCHEDULELib.ScrollBarsEnum.exNoScroll;
```

JavaScript

```
<OBJECT classid="clsid:9B09E13D-7A88-4299-9DBE-383380435377"
id="Schedule1"></OBJECT>
<SCRIPT LANGUAGE="JScript">
Schedule1.OnResizeControl = 768;
Schedule1.ScrollBars = 0;
</SCRIPT>
```

C# for /COM

```
axSchedule1.OnResizeControl =
EXSCHEDULELib.OnResizeControlEnum.exHideSplitter |
```

```
EXSCHEDULELib.OnResizeControlEnum.exChangePanels;
axSchedule1.ScrollBars = EXSCHEDULELib.ScrollBarsEnum.exNoScroll;
```

X++ (Dynamics Ax 2009)

```
public void init()
{
;
super();
exschedule1.OnResizeControl(768/*exHideSplitter | exChangePanels*/);
exschedule1.ScrollBars(0/*exNoScroll*/);
}
```

Delphi 8 (.NET only)

```
with AxSchedule1 do
begin
    OnResizeControl :=
Integer(EXSCHEDULELib.OnResizeControlEnum.exHideSplitter) Or
Integer(EXSCHEDULELib.OnResizeControlEnum.exChangePanels);
    ScrollBars := EXSCHEDULELib.ScrollBarsEnum.exNoScroll;
end
```

Delphi (standard)

```
with Schedule1 do
begin
    OnResizeControl := Integer(EXSCHEDULELib_TLB.exHideSplitter) Or
Integer(EXSCHEDULELib_TLB.exChangePanels);
    ScrollBars := EXSCHEDULELib_TLB.exNoScroll;
end
```

VFP

with thisform.Schedule1 .**OnResizeControl** = 768

```
.ScrollBars = 0
```

endwith

dBASE Plus

local oSchedule

oSchedule = form.Activex1.nativeObject oSchedule.**OnResizeControl** = 768 /*exHideSplitter | exChangePanels*/ oSchedule.ScrollBars = 0

XBasic (Alpha Five)

Dim oSchedule as P

oSchedule = topparent:CONTROL_ACTIVEX1.activex oSchedule.**OnResizeControl** = 768 **'exHideSplitter + exChangePanels** oSchedule.ScrollBars = 0

Visual Objects

oDCOCX_Exontrol1:**OnResizeControl** := exHideSplitter | exChangePanels oDCOCX_Exontrol1:ScrollBars := exNoScroll

PowerBuilder

```
OleObject oSchedule
oSchedule = ole_1.Object
oSchedule.OnResizeControl = 768 /*exHideSplitter | exChangePanels*/
oSchedule.ScrollBars = 0
```

property Schedule.PaneMinWidth(Right as Boolean) as Long

Specifies the minimum width for the left or right panel.

Туре	Description
Right as Boolean	A Boolean expression that defines the panel whose width is requested.
Long	A long expression that specifies the minimum width of the giving panel.

The PaneMinWidth property specifies the minimum width for the panel. The <u>PaneWidth</u> property indicates the width of the left or right panel.

Also, you can use the <u>OnResizeControl</u> property to specify one of the followings:

- **auto hide** the calendar panel. Ability to hide the calendar section while the cursor is not in it (OnResizeControlEnum.exResizePanelRight Or OnResizeControlEnum.exCalendarFit Or OnResizeControlEnum.exCalendarAutoHide).
- hide completely the calendar section (exHideSplitter)
- specify the **alignment** of the calendar, as on the left or right side of the schedule view (OnResizeControlEnum.exChangePanels Or OnResizeControlEnum.exCalendarFit)
- full or **partially** view of the calendar panel (exResizePanelRight)
- disabling the control's vertical split bar (so user can not resize the fixed panel) (OnResizeControlEnum.exResizePanelRight Or OnResizeControlEnum.exDisableSplitter Or OnResizeControlEnum.exCalendarFit)

property Schedule.PaneWidth(Right as Boolean) as Long

Specifies the width for the left or right panel.

Туре	Description
Right as Boolean	A Boolean expression that defines the panel whose width is requested.
Long	A long expression that specifies the width of the giving panel.

The PaneWidth property indicates the width of the left or right panel. The <u>PaneMinWidth</u> property specifies the minimum width for the panel.

Also, you can use the <u>OnResizeControl</u> property to specify one of the followings:

- auto hide the calendar panel. Ability to hide the calendar section while the cursor is not in it (OnResizeControlEnum.exResizePanelRight Or OnResizeControlEnum.exCalendarFit Or OnResizeControlEnum.exCalendarAutoHide).
- hide completely the calendar section (exHideSplitter)
- specify the **alignment** of the calendar, as on the left or right side of the schedule view (OnResizeControlEnum.exChangePanels Or OnResizeControlEnum.exCalendarFit)
- full or **partially** view of the calendar panel (exResizePanelRight)
- disabling the control's vertical split bar (so user can not resize the fixed panel) (OnResizeControlEnum.exResizePanelRight Or OnResizeControlEnum.exDisableSplitter Or OnResizeControlEnum.exCalendarFit)

property Schedule.Picture as IPictureDisp

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

Туре	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 <u>PictureDisplay</u> property to determine how the picture is displayed on the control's background. The <u>BackColor</u> property specifies a solid color to be shown on the control's background. The <u>Background</u>(exCalendarBackColor) property changes the calendar's panel back color if not-zero. You can use the <u>EventsTransparent</u> property to show the events in the control with a transparent color.

The following screen shot shows the control with a picture on its background (tiled):



The <u>ExPictures</u> collection holds a collection of icons, pictures that can be displayed using the <u>Pictures</u> and <u>ExtraPictures</u> properties of the Event object. The <u>Images</u> method loads icons to the control, <u>HTMLPicture</u> assigns a key to a picture object. The ExPictures collection can be accessed through the <u>Pictures</u> property of the control.

The control can display icons, pictures several ways as follows:

 Using the HTML tag, if used in any label properties such as LongLabel or ExtraLabel property. The ShortLabel property can NOT display images or HTML font attributes. For instance, this option can be used to display a default icon or picture for all events in the control using the DefaultEventLongLabel property. Also, the TAG can be used to any label or caption property that supports HTML format.

- Using the <u>Pictures</u> property of the Event object, which indicates a list of pictures from Pictures collection to be displayed on the event's body. The <u>PicturesAlign</u> property indicates the alignment of the pictures relative to the borders of the event. The hand cursor is shown, if the ExPicture.<u>ShowCursorHand</u> property is set, when the cursor hovers the picture. The <u>PictureClick</u> event is fired if the user clicks a picture on event's picture. You can use the <u>PictureFromPoint</u> property to get the identifier of the picture from the cursor.
- Using the <u>ExtraPictures</u> property of the Event object, which indicates a list of pictures from Pictures collection to be displayed on the event's body. The <u>ExtraPicturesAlign</u> property indicates the alignment of the pictures relative to the borders of the event. The hand cursor is shown, if the ExPicture.<u>ShowCursorHand</u> property is set, when the cursor hovers the picture. The <u>PictureClick</u> event is fired if the user clicks a picture on event's picture. You can use the <u>PictureFromPoint</u> property to get the identifier of the picture from the cursor.

property Schedule.PictureDisplay as PictureDisplayEnum

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

Туре	Description
<u>PictureDisplayEnum</u>	A PictureDisplayEnum expression that specifies how the picture is displayed on the control's background.

The control uses the PictureDisplay property to determine how the picture is displayed on the control's background. By default, the control has no picture associated. Use the <u>Picture</u> property to display a picture on the control's background. The <u>BackColor</u> property specifies a solid color to be shown on the control's background. The <u>Background(</u> exCalendarBackColor) property changes the calendar's panel back color if not-zero. You can use the <u>EventsTransparent</u> property to show the events in the control with a transparent color. The <u>PicturesAlign</u> / <u>ExtraPicturesAlign</u> property indicates the alignment of the pictures relative to the borders of the event.

The following screen shot shows the control with a picture on its background (LowerRight):



property Schedule.PictureFromPoint (X as OLE_XPOS_PIXELS, Y as OLE_YPOS_PIXELS) as String

Retrieves the identifier of the picture from the point (Event.Pictures or Event.ExtraPictures).

Туре	Description
X as OLE_XPOS_PIXELS	A single that specifies the current X location of the mouse pointer. The x values is always expressed in client coordinates.
Y as OLE_YPOS_PIXELS	A single that specifies the current Y location of the mouse pointer. The y values is always expressed in client coordinates.
String	A String expression that indicates the key of the picture from the cursor.

The <u>MouseMove</u> event is generated continually as the mouse pointer moves across objects. During the MouseMove event you can call the <u>ShowToolTip</u> method to display any custom tooltip. During the <u>Click</u> or <u>RClick</u> event you can get an UI part of the control using one of the following properties. *All ...FromPoint properties can be use such as ...FromPoint(-1,-1) to get the UI part of the control from the current mouse position, in other words, you do not have to pass any X, Y coordinates.*

You can get UI parts from the cursor, using any of the following ... FromPoint properties:

- Calendar. <u>DateFromPoint</u> property gets the date from the cursor in the Calendar panel.
- <u>AnchorFromPoint</u> property gets the anchor (<a>) element from the cursor.
- <u>DateTimeFromPoint</u> property gets the date/time from the cursor, in the Schedule panel.
- <u>TimeFromPoint</u> property gets the time from the cursor, in the Schedule panel.
- EventFromPoint property gets the Event object from the cursor.
- <u>GroupFromPoint</u> property retrieves the <u>Group</u> object from the cursor. Use the GroupHeaderFromPoint property to the Group object when cursor hovers the group's header.
- <u>GroupHeaderFromPoint</u> property retrieves the <u>Group</u> object from the cursor, when the cursor hovers just the group's header.
- <u>MarkTimeFromPoint</u> property gets the <u>MarkTime</u> object from the cursor. A MarkTime object is identified by a date/time where it is displaying.
- <u>MarkZoneFromPoint</u> property gets the <u>MarkZone</u> object from the cursor. A MarkZone object is identified by a starting date/time and ending date/time where it is displaying.
- <u>NonworkingTimeFromPoint</u> property indicates the <u>NonworkingTime</u> object from the cursor.
- <u>PictureFromPoint</u> property indicates the key of the picture from the cursor.

• <u>TimeScaleFromPoint</u> property returns the <u>TimeScale</u> object from the cursor.

property Schedule.Pictures as ExPictures

Gets the Pictures collection of the scheduler.

Туре	Description
<u>ExPictures</u>	The ExPictures object that holds the icons and pictures that may be shown on the event's body using the <u>Pictures</u> and <u>ExtraPictures</u> properties of the Event object.

The ExPictures collection holds a collection of icons, pictures that can be displayed using the <u>Pictures</u> and <u>ExtraPictures</u> properties of the Event object. The <u>Images</u> method loads icons to the control, <u>HTMLPicture</u> assigns a key to a picture object. The ExPictures collection can be accessed through the <u>Pictures</u> property of the control.

The control can display icons, pictures several ways as follows:

- Using the HTML tag, if used in any label properties such as LongLabel or ExtraLabel property. The ShortLabel property can NOT display images or HTML font attributes. For instance, this option can be used to display a default icon or picture for all events in the control using the DefaultEventLongLabel property. Also, the TAG can be used to any label or caption property that supports HTML format.
- Using the <u>Pictures</u> property of the Event object, which indicates a list of pictures from Pictures collection to be displayed on the event's body. The <u>PicturesAlign</u> property indicates the alignment of the pictures relative to the borders of the event. The hand cursor is shown, if the ExPicture.<u>ShowCursorHand</u> property is set, when the cursor hovers the picture. The <u>PictureClick</u> event is fired if the user clicks a picture on event's picture. You can use the <u>PictureFromPoint</u> property to get the identifier of the picture from the cursor.
- Using the <u>ExtraPictures</u> property of the Event object, which indicates a list of pictures from Pictures collection to be displayed on the event's body. The <u>ExtraPicturesAlign</u> property indicates the alignment of the pictures relative to the borders of the event. The hand cursor is shown, if the ExPicture.<u>ShowCursorHand</u> property is set, when the cursor hovers the picture. The <u>PictureClick</u> event is fired if the user clicks a picture on event's picture. You can use the <u>PictureFromPoint</u> property to get the identifier of the picture from the cursor.

method Schedule.Redo ()

Redoes the next action in the control's Redo queue.

Туре

Description

The Redo redoes the next action in the control's redo queue. The <u>AllowUndoRedo</u> property enables or disables the Undo/Redo feature. The <u>CanRedo</u> method indicates whether the control can perform a Redo operation. The <u>Undo</u> method undoes the last control operation. The <u>UndoRedoQueueLength</u> property gets or sets the maximum number of Undo/Redo actions that may be stored to the control's queue, or in other words how many operations the control's Undo/Redo manager may store.

The records of the Undo/Redo queue may contain actions in the following format:

- "AddEvent; EVENTID", indicates that a new calendar-event has been created
- "RemoveEvent; EVENTID", indicates that an calendar-event has been removed
- "MoveEvent; EVENTID", indicates that an calendar-event has been moved or resized
- "UpdateEvent; EVENTID", indicates that one or more properties of the calendar-event has been updated, using the <u>StartUpdateEvent</u> / <u>EndUpdateEvent</u> methods

Also, the Undo/Redo queue may include:

- "StartBlock", specifies that a block of operations begins (initiated by <u>StartBlockUndoRedo</u> method)
- "EndBlock", specifies that a block of operations ends (initiated by EndBlockUndoRedo method)

The LayoutStartChanging(exUndo/exRedo) / LayoutEndChanging(exUndo/exRedo) event notifies your application whenever an Undo/Redo operation is performed. The UndoListAction property lists the Undo actions that can be performed in the control. The RedoListAction property lists the Redo actions that can be performed in the control. Use the UndoRemoveAction method to remove the last actions from the undo queue. The RedoRemoveAction method removes the first action to be performed if the Redo method is invoked.

property Schedule.RedoListAction ([Action as Variant], [Count as Variant]) as String

Lists the Redo actions that can be performed on the control.

Туре	Description
Action as Variant	[optional] A long expression that specifies the action being listed. If missing or -1, all actions are listed.
	The Action parameter can be one of the following:
	 exUndoRedoAddEvent(13) ~ "AddEvent;EVENTID", indicates that a new calendar-event has been created exUndoRedoRemoveEvent(14) ~ "RemoveEvent;EVENTID", indicates that an calendar-event has been removed exUndoRedoMoveEvent(15) ~ "MoveEvent;EVENTID", indicates that an calendar-event has been moved or resized exUndoRedoUpdateEvent(16) ~ "UpdateEvent;EVENTID", indicates that one or more properties of the calendar-event has been updated, using the <u>StartUpdateEvent</u> / EndUpdateEvent methods
	For instance, RedoListAction(12) shows only AddEvent actions in the redo stack.
Count as Variant	[optional] A long expression that indicates the number of actions being listed. If missing or -1, all actions are listed. For instance, RedoListAction(12,1) shows only the last AddEvent action being added to the redo stack
String	A String expression that lists the Redo actions that may be performed.

The RedoListAction property lists the Redo actions that can be performed in the control. The <u>AllowUndoRedo</u> property enables or disables the Undo/Redo feature. The <u>UndoListAction</u> property lists the Undo actions that can be performed in the control. Use the <u>UndoRemoveAction</u> method to remove the last actions from the undo queue. The <u>RedoRemoveAction</u> method removes the first action to be performed if the Redo method is invoked. The <u>LayoutStartChanging(exUndo/exRedo) / LayoutEndChanging(exUndo/exRedo)</u> event notifies your application whenever an Undo/Redo operation is performed. The records of the Undo/Redo queue may contain actions in the following format:

- "AddEvent; EVENTID", indicates that a new calendar-event has been created
- "RemoveEvent; EVENTID", indicates that an calendar-event has been removed
- "MoveEvent; EVENTID", indicates that an calendar-event has been moved or resized
- "UpdateEvent; EVENTID", indicates that one or more properties of the calendar-event has been updated, using the <u>StartUpdateEvent</u> / <u>EndUpdateEvent</u> methods

Also, the Undo/Redo queue may include:

- "StartBlock", specifies that a block of operations begins (initiated by <u>StartBlockUndoRedo</u> method)
- "EndBlock", specifies that a block of operations ends (initiated by <u>EndBlockUndoRedo</u> method)

Here's a sample how the result of RedoListAction method looks like:

AddEvent;1 UpdateEvent;1 AddEvent;2 UpdateEvent;2 MoveEvent;8 StartBlock MoveEvent;3 AddEvent;3 EndBlock

method Schedule.RedoRemoveAction ([Action as Variant], [Count as Variant])

Removes the last redo actions that can be performed on the control.

Туре	Description
	[optional] A long expression that specifies the action being remove. If missing or -1, all actions are removed. The Action parameter can be one of the following:
Action as Variant	 exUndoRedoAddEvent(13) ~ "AddEvent;EVENTID", indicates that a new calendar-event has been created exUndoRedoRemoveEvent(14) ~ "RemoveEvent;EVENTID", indicates that an calendar-event has been removed exUndoRedoMoveEvent(15) ~ "MoveEvent;EVENTID", indicates that an calendar- event has been moved or resized exUndoRedoUpdateEvent(16) ~ "UpdateEvent;EVENTID", indicates that one or more properties of the calendar-event has been updated, using the <u>StartUpdateEvent</u> / <u>EndUpdateEvent</u> methods
	Addevent actions from the redo stack.
Count as Variant	[optional] A long expression that indicates the number of actions to remove. If missing or -1, all actions are removed. For instance, RedoRemoveAction(12,1) removes only the last AddEvent action from the redo stack

The RedoRemoveAction method removes the first action to be performed if the Redo method is invoked. Use the RedoRemoveAction() (with no parameters) to remove all redo actions. Use the <u>UndoRemoveAction</u> method to remove the last action from the undo queue. The <u>AllowUndoRedo</u> property enables or disables the Undo/Redo feature. The <u>UndoListAction</u> property lists the Undo actions that can be performed in the control. The <u>RedoListAction</u> property lists the Redo actions that can be performed in the control. The <u>LayoutStartChanging</u>(exUndo/exRedo) / <u>LayoutEndChanging</u>(exUndo/exRedo) event notifies your application whenever an Undo/Redo operation is performed.

The records of the Undo/Redo queue may contain actions in the following format:

- "AddEvent; EVENTID", indicates that a new calendar-event has been created
- "RemoveEvent; EVENTID", indicates that an calendar-event has been removed
- "**MoveEvent**; EVENTID", indicates that an calendar-event has been moved or resized
- "UpdateEvent; EVENTID", indicates that one or more properties of the calendar-event has been updated, using the <u>StartUpdateEvent</u> / <u>EndUpdateEvent</u> methods

Also, the Undo/Redo queue may include:

- "StartBlock", specifies that a block of operations begins (initiated by <u>StartBlockUndoRedo</u> method)
- "EndBlock", specifies that a block of operations ends (initiated by EndBlockUndoRedo method)

method Schedule.Refresh ()

Refreshes the control.

Туре

Description

The Refresh method refreshes the controls content. You can use the <u>BeginUpdate</u> and <u>EndUpdate</u> methods to increase the speed of loading your events, by preventing painting the control when it suffers any change.

method Schedule.RemoveSelection ()

Removes the selected events.

Туре

Description

method Schedule.Replacelcon ([lcon as Variant], [Index as Variant])

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

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

Use the ReplaceIcon property to add, remove or replace an icon in the control's images

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

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

i = ExSchedule1.ReplaceIcon(LoadPicture("d:\icons\help.ico").Handle), i specifies the index where the icon is added

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

i = ExSchedule1.ReplaceIcon(LoadPicture("d:\icons\help.ico").Handle, 0), i is zero, so the first icon is replaced.

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

ExSchedule1.ReplaceIcon 0, i, where i specifies the index of icon removed.

The following VB clears the control's icons collection:

ExSchedule1.ReplaceIcon 0, -1

The <u>Images</u> method loads icons to the control, <u>HTMLPicture</u> assigns a key to a picture object, and the <u>Pictures</u> collection handles the identifiers of the pictures that can be used in the Pictures or <u>ExtraPictures</u> properties.

In order to display an icon or a picture in the control you need first to load the icons or the pictures you plan to display, using the <u>Images</u> method, <u>HTMLPicture</u>, or <u>Add</u> method of the ExPictures collection. The Images collection can display only 16x16 icons, while the <u>HTMLPicture</u>, or <u>Add</u> method can load and display custom sized pictures. The <u>Width/Height</u> property specifies the width and height of the picture to be displayed in the event's body.

The event can display icons, pictures several ways as follows:

- Using the HTML tag, if used in any label properties such as LongLabel or ExtraLabel property. The ShortLabel property can NOT display images or HTML font attributes. For instance, this option can be used to display a default icon or picture for all events in the control using the DefaultEventLongLabel property. Also, the TAG can be used to any label or caption property that supports HTML format.
- Using the <u>Pictures</u> property of the Event object, which indicates a list of pictures from Pictures collection to be displayed on the event's body. The <u>PicturesAlign</u> property indicates the alignment of the pictures relative to the borders of the event. The hand cursor is shown, if the ExPicture.<u>ShowCursorHand</u> property is set, when the cursor

hovers the picture. The <u>PictureClick</u> event is fired if the user clicks a picture on event's picture. You can use the <u>PictureFromPoint</u> property to get the identifier of the picture from the cursor.

Using the <u>ExtraPictures</u> property of the Event object, which indicates a list of pictures from Pictures collection to be displayed on the event's body. The <u>ExtraPicturesAlign</u> property indicates the alignment of the pictures relative to the borders of the event. The hand cursor is shown, if the ExPicture.<u>ShowCursorHand</u> property is set, when the cursor hovers the picture. The <u>PictureClick</u> event is fired if the user clicks a picture on event's picture. You can use the <u>PictureFromPoint</u> property to get the identifier of the picture from the cursor.

The <u>Pictures</u> and <u>ExtraPictures</u> may display one or more pictures at the time. The , character indicates the separator of pictures in the same line, while the *I* character divides the lines to show the pictures. For instance, "1,2" displays icon with the index 1 and 2 on the same line, while the "1/2,pic1" displays the first icon on the first line, the second icon and the picture pic1 on the second line.

method Schedule.SaveXML (Destination as Variant)

Saves the control's content as XML document to the specified location, using the MSXML parser.

Туре	Description
	This object can represent a file name, reference to a string member, an XML document object, or a custom object that supports persistence as follows:
	• String - Specifies the file name. Note that this must be a file name, rather than a URL. The file is created if necessary and the contents are entirely replaced with the contents of the saved document. For example:
	<pre>Schedule1.SaveXML("sample.xml")</pre>
	 Reference to a String member - Saves the control's content to the string member. Note that the string member must be empty, before calling the SaveXML method. For example:
	Dim s As String Schedule1.SaveXML s
	In VB.NET for /NET assembly, you should call such as :
Destination as Variant	Dim s As String = String.Empty Exschedule1.SaveXML(s)
	In C# for /NET assembly, you should call such as :
	<pre>string s = string.Empty; exschedule1.SaveXML(ref s);</pre>
	 XML Document Object. For example:
	Dim xmldoc as Object Set xmldoc = CreateObject("MSXML.DOMDocument") Schedule1.SaveXML(xmldoc)
	 Custom object supporting persistence - Any other custom COM object that supports QueryInterface for IStream, IPersistStream, or IPersistStreamInit can also be provided here and the document will be saved accordingly. In the IStream case, the IStream::Write

method will be called as it saves the document; in the **IPersistStream** case, **IPersistStream::Load** will be called with an **IStream** that supports the **Read**, **Seek**, and **Stat** methods.

Return	Description
Boolean	A Boolen expression that specifies whether saving the XML document was ok.

The SaveXML method uses the MSXML (MSXML.DOMDocument, XML DOM Document) parser to save the control's data in XML documents. The <u>LoadXML</u> method loads XML documents being created with SaveXML method. The <u>Copy</u>, <u>CopyTo</u> methods copies the control's content in EMF format.

The <u>XML format</u> looks like follows:

 - <Content Author Component Version> <DateFormat Separator Short Long /> <TimeFormat Separator Format AM PM /> <NumberFormat Decimal /> -<Calendar WeekDays MonthNames AMPM FirstWeekDay NonworkingDays ShortDateFormat LongDateFormat ShortTimeFormat LongTimeFormat GroupHighlightEven ShowHighlightEvent DisableZoneFormat MinDate MaxDate ShowNonMonthDays Date SingleSel HideSel ... > - < Selection > <Date Value .../> </Selection> <HighlightEvent Bold Italic Underline StrikeOut FontSize .../> </Calendar> - <TimeScales DayStartTime DayEndTime> <TimeScale TimeZone AlignLeft MinWidth MaxWidth Width AllowResize MajorRuler MajorLabel MajorPlainText MinorRuler MinorLabel MinorPlainText Visible Caption CaptionAlign ToolTip UserData MinorLabelColor .../> </TimeScales> - <Groups ApplyGroupingColors DisplayGroupingButton GroupHighlightEvent ShowGroupingEvents> -<Group ID Visible Caption Title ToolTip Alignment UserData> <CalendarHighlightEvent Bold Italic Underline StrikeOut FontSize ... /> <ScheduleHighlightEvent Bold Italic Underline StrikeOut FontSize ... /> </Group> </Groups> - <NonworkingPatterns> <NonworkingPattern ID PatternType PatternColor PatternFrameColor .../> </NonworkingPatterns> -<NonworkingTimes ShowNonworkingTime> <NonworkingTime Expression StartTime EndTime IDNonworkingPattern .../> </NonworkingTimes> -<MarkZones> <MarkZone Key Start End ShortLabel LongLabel GroupID PatternType PatternColor PatternFrameColor ... /> </MarkZones> -<MarkTimes ShowMarkTime> <MarkTime Key Time Line TimeScaleLine LineColor TimeScaleLineColor Label TimeScaleLabel LabelAlign TimeScaleLabelAlign Movable StatusEventBackColor UserData ... /> </MarkTimes> - <Pictures ShowEventPictures> <Picture Key Content Width Height ShowHandCursor Enabled ... /> </Pictures> <Icons Content/> -<Events HeaderDayLongLabel HeaderDayShortLabel CreateEventLabel CreateEventLabelAlign UpdateEventsLabel UpdateEventsLabelAlign DefaultEventLongLabel DefaultEventTooltip DefaultEventPaddingLeft DefaultEventPaddingTop DefaultEventPaddingRight DefaultEventPaddingBottom ...> <Event Start End GroupID Selectable ShowStatus ShortLabel LongLabel LabelAlign ExtraLabelAlign Pictures ... />

</Events> </Content>
property Schedule.ScrollBars as ScrollBarsEnum

Returns or sets a value that determines whether the control has horizontal and/or vertical scroll bars.

Туре	Description
<u>ScrollBarsEnum</u>	A ScrollBarsEnum expression that identifies which scroll bars are visible.

Use the ScrollBars property to disable the control's scroll bars. By default, the ScrollBars property is exBoth, so both scroll bars are used if necessarily. For instance, if the ScrollBars property is exNone the control displays no scroll bars. Use the ScrollOrderParts property to customize the order of the buttons in the scroll bar.

property Schedule.ScrollButtonHeight as Long

Specifies the height of the button in the vertical scrollbar.

Туре	Description
Long	A long expression that defines the height of the button in the vertical scroll bar.

By default, the ScrollButtonHeight property is -1. If the ScrollButtonHeight property is -1, the control uses the default height (from the system) for the buttons in the vertical scroll bar. Use the <u>ScrollButtonWidth</u> property to specify the width of the buttons in the horizontal scroll bar. Use the <u>ScrollWidth</u> property to specify the width of the vertical scroll bar. Use the <u>ScrollBars</u> property to specify which scroll bar is visible or hidden in the control. Use the <u>ScrollHeight</u> property to specify the height of the horizontal scroll bar. Use the <u>ScrollPartVisible</u> property to specify the visible parts in the control's scroll bar. Use the <u>ScrollThumbSize</u> property to define a fixed size for the scrollbar's thumb.

property Schedule.ScrollButtonWidth as Long

Specifies the width of the button in the horizontal scrollbar.

Туре	Description
Long	A long expression that defines the width of the button in the horizontal scroll bar.

By default, the ScrollButtonWidth property is -1. If the ScrollButtonWidth property is -1, the control uses the default width (from the system) for the buttons in the horizontal scroll bar. Use the <u>ScrollButtonHeight</u> property to specify the height of the buttons in the vertical scroll bar. Use the <u>ScrollWidth</u> property to specify the width of the vertical scroll bar. Use the <u>ScrollBars</u> property to specify which scroll bar is visible or hidden in the control. Use the <u>ScrollHeight</u> property to specify the height of the horizontal scroll bar. Use the <u>ScrollPartVisible</u> property to specify the visible parts in the control's scroll bar. Use the <u>ScrollThumbSize</u> property to define a fixed size for the scrollbar's thumb.

property Schedule.ScrollFont (ScrollBar as ScrollBarEnum) as IFontDisp

Retrieves or sets the scrollbar's font.

Туре	Description
ScrollBar as <u>ScrollBarEnum</u>	A ScrollBarEnum expression that indicates the vertical or the horizontal scroll bar.
IFontDisp	A Font object

Use the ScrollFont property to specify the font in the control's scroll bar. Use the <u>ScrolPartCaption</u> property to specify the caption of the scroll's part. Use the <u>ScrollPartVisible</u> property to add or remove buttons/parts in the control's scrollbar. By default, when a part becomes visible, the <u>ScrollPartEnable</u> property is automatically called, so the parts becomes enabled. Use the <u>ScrollPartEnable</u> property to specify enable or disable parts in the control's scrollbar. Use the <u>ScrollBars</u> property to specify the visible scrollbars in the control. The control fires the <u>ScrollButtonClick</u> event when the user clicks a part of the scroll bar.

property Schedule.ScrollHeight as Long

Specifies the height of the horizontal scrollbar.

Туре	Description
Long	A long expression that defines the height of the horizontal scroll bar.

By default, the ScrollHeight property is -1. If the ScrollHeight property is -1, the control uses the default height of the horizontal scroll bar from the system. Use the ScrollHeight property to specify the height of the horizontal scroll bar. Use the <u>ScrollBars</u> property to specify which scroll bar is visible or hidden in the control. Use the <u>ScrollButtonWidth</u> property to specify the width of the buttons in the horizontal scroll bar. Use the <u>ScrollButtonWidth</u> property to specify the width of the vertical scroll bar. Use the <u>ScrollButtonHeight</u> property to specify the height of the buttons in the vertical scroll bar. Use the <u>ScrollPartVisible</u> property to specify the visible parts in the control's scroll bar. Use the <u>ScrollThumbSize</u> property to define a fixed size for the scrollbar's thumb.

property Schedule.ScrollOrderParts(ScrollBar as ScrollBarEnum) as String

Specifies the order of the buttons in the scroll bar.

Туре	Description
ScrollBar as <u>ScrollBarEnum</u>	A ScrollBar expression that indicates the scrollbar where the order of buttons is displayed.
String	A String expression that indicates the order of the buttons in the scroll bar. The list includes expressions like I, I1,, I5, t, r, r1,, r6 separated by comma, each expression indicating a <u>part</u> of the scroll bar, and its position indicating the displaying order.

Use the ScrollOrderParts to customize the order of the buttons in the scroll bar. By default, the ScrollOrderParts property is empty. If the ScrollOrderParts property is empty the default order of the buttons in the scroll bar are displayed like follows:

L1 L2 L3 L4 L5 < MI R2 R3 R4 R5 R6

so, the order of the parts is: I1, I2, I3, I4, I5, I, t, r, r1, r2, r3, r4, r5 and r6. Use the <u>ScrollPartVisible</u> to specify whether a button in the scrollbar is visible or hidden. Use the <u>ScrollPartEnable</u> property to enable or disable a button in the scroll bar. Use the <u>ScrollPartCaption</u> property to assign a caption to a button in the scroll bar.

Use the ScrollOrderParts property to change the order of the buttons in the scroll bar. For instance, "I,r,t,I1,r1" puts the left and right buttons to the left of the thumb area, and the I1 and r1 buttons right after the thumb area. If the parts are not specified in the ScrollOrderParts property, automatically they are added to the end.

< > III R1 L2 L3 L4 L5 R2 R3 R4 R5 R6

The list of supported literals in the ScrollOrderParts property is:

- I for exLeftBPart, (<) The left or top button.
- I1 for exLeftB1Part, (L1) The first additional button, in the left or top area.
- 12 for exLeftB2Part, (L2) The second additional button, in the left or top area.
- **I3** for exLeftB3Part, (L3) The third additional button, in the left or top area.
- I4 for exLeftB4Part, (L4) The forth additional button, in the left or top area.
- 15 for exLeftB5Part, (L5) The fifth additional button, in the left or top area.
- **t** for exLowerBackPart, exThumbPart and exUpperBackPart, The union between the exLowerBackPart and the exUpperBackPart parts.
- **r** for exRightBPart, (>) The right or down button.

- **r1** for exRightB1Part, (R1) The first additional button in the right or down side.
- **r2** for exRightB2Part, (R2) The second additional button in the right or down side.
- **r3** for exRightB3Part, (R3) The third additional button in the right or down side.
- **r4** for exRightB4Part, (R4) The forth additional button in the right or down side.
- **r5** for exRightB5Part, (R5) The fifth additional button in the right or down side.
- r6 for exRightB6Part, (R6) The sixth additional button in the right or down side.

Any other literal between commas is ignored. If duplicate literals are found, the second is ignored, and so on. For instance, "t,l,r" indicates that the left/top and right/bottom buttons are displayed right/bottom after the thumb area.

property Schedule.ScrollPartCaption(ScrollBar as ScrollBarEnum, Part as ScrollPartEnum) as String

Specifies the caption being displayed on the specified scroll part.

Туре	Description
ScrollBar as <u>ScrollBarEnum</u>	A ScrollBar expression that indicates the scrollbar where the caption is displayed.
Part as <u>ScrollPartEnum</u>	A ScrollPartEnum expression that specifies the parts of the scroll where the text is displayed
String	A String expression that specifies the caption being displayed on the part of the scroll bar.

Use the ScrolPartCaption property to specify the caption of the scroll's part. Use the ScrollPartVisible property to add or remove buttons/parts in the control's scrollbar. By default, when a part becomes visible, the ScrollPartEnable property is automatically called, so the parts becomes enabled. Use the ScrollPartEnable property to specify enable or disable parts in the control's scrollbar. Use the ScrollBars property to specify the visible scrollbars in the control. The control fires the ScrollButtonClick event when the user clicks a part of the scroll bar. Use the ScrollFont property to specify the font in the control's scroll bar. Use the ScrollPartEnable or disable parts in the control fires the ScrollButtonClick event when the user clicks a part of the scroll bar. Use the ScrollFont property to specify the font in the control's scroll bar. Use the ScrollPartCaptionAlignment property specifies the alignment of the caption in the part of the scroll bar.



By default, the following parts are shown:

- exLeftBPart (the left or up button of the control)
- exLowerBackPart (the part between the left/up button and the thumb part of the control)
- exThumbPart (the thumb/scrollbox part)
- exUpperBackPart (the part between the the thumb and the right/down button of the control)
- exRightBPart (the right or down button of the control)

The following VB sample adds up and down additional buttons to the control's vertical scroll bar :

With Schedule1

```
.BeginUpdate

.ScrollBars = exDisableBoth

.ScrollPartVisible(exVScroll, exLeftB1Part Or exRightB1Part) = True

.ScrollPartCaption(exVScroll, exLeftB1Part) = "<img></img>1"

.ScrollPartCaption(exVScroll, exRightB1Part) = "<img></img>2"

.EndUpdate

End With
```

The following VB.NET sample adds up and down additional buttons to the control's vertical scroll bar :

```
With AxSchedule1
.BeginUpdate()
.ScrollBars = EXSCHEDULELib.ScrollBarsEnum.exDisableBoth
.set_ScrollPartVisible(EXSCHEDULELib.ScrollBarEnum.exVScroll,
EXSCHEDULELib.ScrollPartEnum.exLeftB1Part Or
EXSCHEDULELib.ScrollPartEnum.exRightB1Part, True)
.set_ScrollPartCaption(EXSCHEDULELib.ScrollBarEnum.exVScroll,
EXSCHEDULELib.ScrollPartEnum.exLeftB1Part, "<img></img>1")
.set_ScrollPartCaption(EXSCHEDULELib.ScrollBarEnum.exVScroll,
EXSCHEDULELib.ScrollPartEnum.exRightB1Part, "<img></img>2")
.EndUpdate()
End With
```

The following C# sample adds up and down additional buttons to the control's vertical scroll bar :

```
axSchedule1.BeginUpdate();
axSchedule1.ScrollBars = EXSCHEDULELib.ScrollBarsEnum.exDisableBoth;
axSchedule1.set_ScrollPartVisible(EXSCHEDULELib.ScrollBarEnum.exVScroll,
EXSCHEDULELib.ScrollPartEnum.exLeftB1Part |
EXSCHEDULELib.ScrollPartEnum.exRightB1Part, true);
axSchedule1.set_ScrollPartCaption(EXSCHEDULELib.ScrollBarEnum.exVScroll,
EXSCHEDULELib.ScrollPartEnum.exLeftB1Part , "<img> </img> 1");
axSchedule1.set_ScrollPartCaption(EXSCHEDULELib.ScrollBarEnum.exVScroll,
EXSCHEDULELib.ScrollPartEnum.exLeftB1Part , "<img> </img> 2");
axSchedule1.EndUpdate();
```

The following C++ sample adds up and down additional buttons to the control's vertical

scroll bar :

```
m_schedule.BeginUpdate();
m_schedule.SetScrollBars( 15 /*exDisableBoth*/ );
m_schedule.SetScrollPartVisible( 0 /*exVScroll*/, 32768 /*exLeftB1Part*/ | 32
/*exRightB1Part*/, TRUE );
m_schedule.SetScrollPartCaption( 0 /*exVScroll*/, 32768 /*exLeftB1Part*/ , _T("<img>
</img>1") );
m_schedule.SetScrollPartCaption( 0 /*exVScroll*/, 32 /*exRightB1Part*/ , _T("<img>
</img>2") );
m_schedule.EndUpdate();
```

The following VFP sample adds up and down additional buttons to the control's vertical scroll bar :

```
With thisform.Schedule1

.BeginUpdate

.ScrollBars = 15

.ScrollPartVisible(0, bitor(32768,32)) = .t.

.ScrollPartCaption(0,32768) = "<img></img>1"

.ScrollPartCaption(0, 32) = "<img></img>2"

.EndUpdate

EndWith
```

```
*** ActiveX Control Event ***
LPARAMETERS scrollpart
```

wait window nowait ltrim(str(scrollpart))

property Schedule.ScrollPartCaptionAlignment(ScrollBar as ScrollBarEnum, Part as ScrollPartEnum) as AlignmentEnum

Specifies the alignment of the caption in the part of the scroll bar.

Туре	Description
ScrollBar as <u>ScrollBarEnum</u>	A ScrollBar expression that indicates the scrollbar where the caption is displayed.
Part as <u>ScrollPartEnum</u>	A ScrollPartEnum expression that specifies the parts of the scroll where the text is displayed
<u>AlignmentEnum</u>	An AlignmentEnum expression that specifies the alignment of the caption in the part of the scrollbar.

The ScrollPartCaptionAlignment property specifies the alignment of the caption in the part of the scroll bar. By default, the caption is centered. Use the <u>ScrolPartCaption</u> property to specify the caption being displayed on specified part of the scroll bar. Use the <u>ScrollPartVisible</u> property to add or remove buttons/parts in the control's scrollbar.

The following VB sample displays "left" aligned to the left on the lower part of the control's horizontal scroll bar, and "right" aligned to the right on the upper part of the control's horizontal scroll bar:

```
With Schedule1
.ScrollPartCaption(exHScroll,exLowerBackPart) = "left"
.ScrollPartCaptionAlignment(exHScroll,exLowerBackPart) = LeftAlignment
.ScrollPartCaption(exHScroll,exUpperBackPart) = "right"
.ScrollPartCaptionAlignment(exHScroll,exUpperBackPart) = RightAlignment
.ColumnAutoResize = False
.Columns.Add 1
.Columns.Add 2
.Columns.Add 3
.Columns.Add 4
End With
```

The following VB.NET sample displays "left" aligned to the left on the lower part of the control's horizontal scroll bar, and "right" aligned to the right on the upper part of the control's horizontal scroll bar:

With AxSchedule1

```
.set_ScrollPartCaption(EXSCHEDULELib.ScrollBarEnum.exHScroll,EXSCHEDULELib.ScrollPartI
```

```
.set_ScrollPartCaptionAlignment(EXSCHEDULELib.ScrollBarEnum.exHScroll,EXSCHEDULELib
.set_ScrollPartCaption(EXSCHEDULELib.ScrollBarEnum.exHScroll,EXSCHEDULELib.ScrollPartI
.set_ScrollPartCaptionAlignment(EXSCHEDULELib.ScrollBarEnum.exHScroll,EXSCHEDULELib
.ColumnAutoResize = False
.Columns.Add 1
.Columns.Add 2
.Columns.Add 3
.Columns.Add 4
End With
```

The following C# sample displays "left" aligned to the left on the lower part of the control's horizontal scroll bar, and "right" aligned to the right on the upper part of the control's horizontal scroll bar:

```
axSchedule 1.set\_ScrollPartCaption (EXSCHEDULELib.ScrollBarEnum.exHScroll, EXSCHEDULELib.ScrollBarEnum.exHScroll, EXSCHEDULELib.ScrollBarEnum.exHScrollBarEnum.exHScrollBarEnum.exHScrollBarEnum.exHScrollBarEnum.exHScrollBarEnum.exHScrollBarEnum.exHScrollBarEnum.exHScrollBarEnum.exHScrollBarEnum.exHScrollBarEnum.exHScrollBarEnum.exHScrollBarEnum.exHScrollBarEnum.exHScrollBarEnum.exHScrollBarEnum.exHScrollBarEnum.exHScrollBarEnum.exHScrollBarEnum.exHScrollBarEnum.exHScrollBarEnum.exHScrollBarEnum.exHScrollBarEnum.exHScrollBarEnum.exHScrollBarEnum.exHScrollBarEnum.exHScrollBarEnum.exHScrollBarEnum.exHScrollBarEnum.exHScrollBarEnum.exHScrollBarEnum.exHScrollBarEnum.exHScrollBarEnum.exHScrollBarEnum.exHScrollBarEnum.exHScrollBarEnum.exHScrollBarEnum.exHScrollBarEnum.exHScrollBarEnum.exHScrol
```

```
axSchedule1.set\_ScrollPartCaptionAlignment (EXSCHEDULELib.ScrollBarEnum.exHScroll, EXSCHEDULELib.ScrollBarEnum.exHScroll, EXSCHEDULELib.ScrollBarEnum.exHScrollBarEnum.exHScrollBarEnum.exHScrollBarEnum.exHScrollBarEnum.exHScrollBarEnum.exHScrollBarEnum.exHScrollBarEnum.exHScrollBarEnum.exHScrollBarEnum.exHScrollBarEnum.exHScrollBarEnum.exHScrollBarEnum.exHScrollBarEnum.exHScrollBarEnum.exHScrollBarEnum.exHScrollBarEnum.exHScrollBarEnum.exHScrollBarEnum.exHScrollBarEnum.exHScrollBarEnum.exHScrollBarEnum.exHScrollBarEnum.exHScrollBarEnum.exHScrollBarEnum.exHScrollBarEnum.exHScrollBarEnum.exHScrollBarEnum.exHScrollBarEnum.exHScrollBarEnum.exHScrollBarEnum.exHScrollBarEnum.exHScrollBarEnum.exHScrollBarEnum.exHScrollBarEnum.exHScrollBarEnum.exHScrollBarEnum.exHScrollBarEnum.exHScrollBarEnum.exHScrollBarEnum.exHScrollBarEnum.exHScrollBarEnum.exHScrollBarEnum.exHScrollBarEnum.exHScrollBarEnum.exHScrollBarEnum.exHScrollBarEnum.exHScrollBarEnum.exHScrollBarEnum.exHScrollBarEnum.exHScrollBarEnum.exHScrollBarEnum.exHScrollBarEnum.exHScrollBarEnum.exHScrollBarEnum.exHScrollBarEnum.exHScrollBarEnum.exHScrollBarEnum.exHScrollBarEnum.exHScrollBarEnum.exHScrollBarEnum.exHScrollBarEnum.exHScrollBarEnum.exHScrollBarEnum.exHScrollBarEnum.exHScrollBarEnum.exHScrollBarEnum.exHScrollBarEnum.exHScrollBarEn
```

```
axSchedule1.set_ScrollPartCaption(EXSCHEDULELib.ScrollBarEnum.exHScroll,EXSCHEDULEL
```

 $axSchedule1.set_ScrollPartCaptionAlignment (EXSCHEDULELib.ScrollBarEnum.exHScroll, EXSCHEDULELib.ScrollBarEnum.exHScroll, EXSCHEDULELib.ScrollBarEnum.exHScrollBarEn$

axSchedule1.ColumnAutoResize = false; axSchedule1.Columns.Add(1.ToString()); axSchedule1.Columns.Add(2.ToString()); axSchedule1.Columns.Add(3.ToString()); axSchedule1.Columns.Add(4.ToString());

The following C++ sample displays "left" aligned to the left on the lower part of the control's horizontal scroll bar, and "right" aligned to the right on the upper part of the control's

horizontal scroll bar:

```
Copy and paste the following directives to your header file as
  it defines the namespace 'EXSCHEDULELib' for the library: 'ExSchedule 1.0 Control
Library'
  #import "ExSchedule.dll"
  using namespace EXSCHEDULELib;
*/
EXSCHEDULELib::ISchedulePtr spSchedule1 = GetDlgItem(IDC_SCHEDULE1)-
>GetControlUnknown();
spSchedule1-
>PutScrollPartCaption(EXSCHEDULELib::exHScroll,EXSCHEDULELib::exLowerBackPart,L"left"
spSchedule1-
> PutScrollPartCaptionAlignment(EXSCHEDULELib::exHScroll,EXSCHEDULELib::exLowerBack
spSchedule1-
> PutScrollPartCaption(EXSCHEDULELib::exHScroll,EXSCHEDULELib::exUpperBackPart,L"righ
spSchedule1-
> PutScrollPartCaptionAlignment(EXSCHEDULELib::exHScroll,EXSCHEDULELib::exUpperBack
spSchedule1->PutColumnAutoResize(VARIANT_FALSE);
spSchedule1->GetColumns()->Add(L"1");
spSchedule1->GetColumns()->Add(L"2");
spSchedule1->GetColumns()->Add(L"3");
spSchedule1->GetColumns()->Add(L"4");
```

The following VFP sample displays "left" aligned to the left on the lower part of the control's horizontal scroll bar, and "right" aligned to the right on the upper part of the control's horizontal scroll bar:

```
with thisform.Schedule1
.ScrollPartCaption(1,512) = "left"
.ScrollPartCaptionAlignment(1,512) = 0
.ScrollPartCaption(1,128) = "right"
```

```
.ScrollPartCaptionAlignment(1,128) = 2
.ColumnAutoResize = .F.
.Columns.Add(1)
.Columns.Add(2)
.Columns.Add(3)
.Columns.Add(4)
endwith
```

property Schedule.ScrollPartEnable(ScrollBar as ScrollBarEnum, Part as ScrollPartEnum) as Boolean

Indicates whether the specified scroll part is enabled or disabled.

Туре	Description
ScrollBar as <u>ScrollBarEnum</u>	A ScrollBar expression that indicates the scrollbar where the part is enabled or disabled.
Part as <u>ScrollPartEnum</u>	A ScrollPartEnum expression that specifies the parts of the scroll bar being enabled or disabled.
Boolean	A Boolean expression that specifies whether the scrollbar's part is enabled or disabled.

By default, when a part becomes visible, the <u>ScrollPartEnable</u> property is automatically called, so the parts becomes enabled. Use the <u>ScrollPartVisible</u> property to add or remove buttons/parts in the control's scrollbar. Use the <u>ScrollPartEnable</u> property to specify enable or disable parts in the control's scrollbar. Use the <u>ScrollBars</u> property to specify the visible scrollbars in the control. Use the <u>ScrollPartCaption</u> property to specify the caption of the scroll's part. The control fires the <u>ScrollButtonClick</u> event when the user clicks a part of the scroll bar. Use the <u>ScrollOrderParts</u> property to customize the order of the buttons in the scroll bar.



property Schedule.ScrollPartVisible(ScrollBar as ScrollBarEnum, Part as ScrollPartEnum) as Boolean

Indicates whether the specified scroll part is visible or hidden.

Туре	Description
ScrollBar as <u>ScrollBarEnum</u>	A ScrollBar expression that indicates the scrollbar where the part is visible or hidden.
Part as <u>ScrollPartEnum</u>	A ScrollPartEnum expression that specifies the parts of the scroll bar being visible
Boolean	A Boolean expression that specifies whether the scrollbar's part is visible or hidden.

Use the ScrollPartVisible property to add or remove buttons/parts in the control's scrollbar. By default, when a part becomes visible, the <u>ScrollPartEnable</u> property is automatically called, so the parts becomes enabled. Use the <u>ScrollPartEnable</u> property to specify enable or disable parts in the control's scrollbar. Use the <u>ScrollBars</u> property to specify the visible scrollbars in the control. Use the <u>ScrolPartCaption</u> property to specify the caption of the scroll's part. The control fires the <u>ScrollButtonClick</u> event when the user clicks a part of the scroll bar. Use the <u>Background</u> property to change the visual appearance for any part in the control's scroll bar. Use the <u>ScrollOrderParts</u> property to customize the order of the buttons in the scroll bar.



By default, the following parts are shown:

- exLeftBPart (the left or up button of the control)
- exLowerBackPart (the part between the left/up button and the thumb part of the control)
- exThumbPart (the thumb/scrollbox part)
- exUpperBackPart (the part between the the thumb and the right/down button of the control)
- exRightBPart (the right or down button of the control)

The following VB sample adds up and down additional buttons to the control's vertical scroll bar :

With Schedule1 .BeginUpdate

```
.ScrollBars = exDisableBoth

.ScrollPartVisible(exVScroll, exLeftB1Part Or exRightB1Part) = True

.ScrollPartCaption(exVScroll, exLeftB1Part) = "<img></img>1"

.ScrollPartCaption(exVScroll, exRightB1Part) = "<img></img>2"

.EndUpdate

End With
```

The following VB.NET sample adds up and down additional buttons to the control's vertical scroll bar :

```
With AxSchedule1
.BeginUpdate()
.ScrollBars = EXSCHEDULELib.ScrollBarsEnum.exDisableBoth
.set_ScrollPartVisible(EXSCHEDULELib.ScrollBarEnum.exVScroll,
EXSCHEDULELib.ScrollPartEnum.exLeftB1Part Or
EXSCHEDULELib.ScrollPartEnum.exRightB1Part, True)
.set_ScrollPartCaption(EXSCHEDULELib.ScrollBarEnum.exVScroll,
EXSCHEDULELib.ScrollPartEnum.exLeftB1Part, "<img></img>1")
.set_ScrollPartCaption(EXSCHEDULELib.ScrollBarEnum.exVScroll,
EXSCHEDULELib.ScrollPartEnum.exRightB1Part, "<img></img>2")
.EndUpdate()
End With
```

The following C# sample adds up and down additional buttons to the control's vertical scroll bar :

```
axSchedule1.BeginUpdate();
axSchedule1.ScrollBars = EXSCHEDULELib.ScrollBarsEnum.exDisableBoth;
axSchedule1.set_ScrollPartVisible(EXSCHEDULELib.ScrollBarEnum.exVScroll,
EXSCHEDULELib.ScrollPartEnum.exLeftB1Part |
EXSCHEDULELib.ScrollPartEnum.exRightB1Part, true);
axSchedule1.set_ScrollPartCaption(EXSCHEDULELib.ScrollBarEnum.exVScroll,
EXSCHEDULELib.ScrollPartEnum.exLeftB1Part, "<img></img>1");
axSchedule1.set_ScrollPartCaption(EXSCHEDULELib.ScrollBarEnum.exVScroll,
EXSCHEDULELib.ScrollPartEnum.exLeftB1Part, "<img></img>1");
axSchedule1.set_ScrollPartCaption(EXSCHEDULELib.ScrollBarEnum.exVScroll,
EXSCHEDULELib.ScrollPartEnum.exRightB1Part, "<img></img>2");
axSchedule1.EndUpdate();
```

The following C++ sample adds up and down additional buttons to the control's vertical scroll bar :

The following VFP sample adds up and down additional buttons to the control's vertical scroll bar :

With thisform.Schedule1 .BeginUpdate .ScrollBars = 15 .ScrollPartVisible(0, bitor(32768,32)) = .t. .ScrollPartCaption(0,32768) = "1" .ScrollPartCaption(0, 32) = "2" .EndUpdate EndWith

*** ActiveX Control Event *** LPARAMETERS scrollpart

wait window nowait ltrim(str(scrollpart))

property Schedule.ScrollThumbSize(ScrollBar as ScrollBarEnum) as Long

Specifies the size of the thumb in the scrollbar.

Туре	Description
ScrollBar as <u>ScrollBarEnum</u>	A ScrollBarEnum expression that indicates the vertical or the horizontal scroll bar.
Long	A long expression that defines the size of the scrollbar's thumb.

Use the ScrollThumbSize property to define a fixed size for the scrollbar's thumb. By default, the ScrollThumbSize property is -1, that makes the control computes automatically the size of the thumb based on the scrollbar's range. If case, use the fixed size for your thumb when you change its visual appearance using the Background(exVSThumb) or Background(exHSThumb) property. Use the ScrollWidth property to specify the width of the vertical scroll bar. Use the ScrollButtonWidth property to specify the width of the horizontal scroll bar. Use the ScrollHeight property to specify the height of the horizontal scroll bar. Use the ScrollPartVisible property to specify the visible parts in the vertical scroll bar. Use the ScrollPartVisible property to specify the visible parts in the control's scroll bar.

property Schedule.ScrollToolTip(ScrollBar as ScrollBarEnum) as String

Specifies the tooltip being shown when the user moves the scroll box.

Туре	Description
ScrollBar as <u>ScrollBarEnum</u>	A ScrollBarEnum expression that indicates the vertical scroll bar or the horizontal scroll bar.
String	A string expression being shown when the user clicks and moves the scrollbar's thumb.

Use the ScrollToolTip property to specify whether the control displays a tooltip when the user clicks and moves the scrollbar's thumb. By default, the ScrollToolTip property is empty. If the ScrollToolTip property is empty, the tooltip is not shown when the user clicks and moves the thumb of the scroll bar. Use the <u>ScrolPartVisible</u> property to specify the parts being visible in the control's scroll bar. Use the <u>ScrollBars</u> property to specify the visible scrollbars in the control.

The following VB sample displays a tooltip when the user clicks and moves the thumb in the control's scroll bar:

Private Sub Schedule1_OffsetChanged(ByVal Horizontal As Boolean, ByVal NewVal As
Long)
If (Not Horizontal) Then
Schedule1.ScrollToolTip(exVScroll) = "Record " & NewVal
End If
End Sub

The following VB.NET sample displays a tooltip when the user clicks and moves the thumb in the control's scroll bar:

Private Sub AxSchedule1_OffsetChanged(ByVal sender As System.Object, ByVal e As AxEXSCHEDULELib._IScheduleEvents_OffsetChangedEvent) Handles AxSchedule1.OffsetChanged If (Not e.horizontal) Then AxSchedule1.set_ScrollToolTip(EXSCHEDULELib.ScrollBarEnum.exVScroll, "Record " & e.newVal.ToString()) End If End Sub

The following C++ sample displays a tooltip when the user clicks and moves the thumb in the control's scroll bar:

```
void OnOffsetChangedSchedule1(BOOL Horizontal, long NewVal)
{
    if ( !Horizontal )
    {
        CString strFormat;
        strFormat.Format( _T("%i"), NewVal );
        m_schedule.SetScrollToolTip( 0, strFormat );
    }
}
```

The following C# sample displays a tooltip when the user clicks and moves the thumb in the control's scroll bar:

```
private void axSchedule1_OffsetChanged(object sender,
AxEXSCHEDULELib._IScheduleEvents_OffsetChangedEvent e)
{
    if ( !e.horizontal )
        axSchedule1.set_ScrollToolTip(EXSCHEDULELib.ScrollBarEnum.exVScroll, "Record " +
    e.newVal.ToString());
```

The following VFP sample displays a tooltip when the user clicks and moves the thumb in the control's scroll bar:

*** ActiveX Control Event *** LPARAMETERS horizontal, newval

If (1 # horizontal) Then
 thisform.Schedule1.ScrollToolTip(0) = "Record " + ltrim(str(newval))
EndIf

property Schedule.ScrollWidth as Long

Specifies the width of the vertical scrollbar.

Туре	Description
Long	A long expression that defines the width of the vertical scroll bar.

By default, the ScrollWidth property is -1. If the ScrollWidth property is -1, the control uses the default width of the vertical scroll bar from the system. Use the ScrollWidth property to specify the width of the vertical scroll bar. Use the <u>ScrollBars</u> property to specify which scroll bar is visible or hidden in the control. Use the <u>ScrollButtonWidth</u> property to specify the width of the buttons in the horizontal scroll bar. Use the <u>ScrollButtonHeight</u> property to specify the height of the horizontal scroll bar. Use the <u>ScrollButtonHeight</u> property to specify the height of the buttons in the vertical scroll bar. Use the <u>ScrollPartVisible</u> property to specify the visible parts in the control's scroll bar. Use the <u>ScrollThumbSize</u> property to define a fixed size for the scrollbar's thumb.

property Schedule.SelCount as Long

Indicates the number of events being selected in the schedule panel.

Туре	Description
Long	A Long expression that specifies the number of selected events.

The SelCount/SelEvent property may be used to retrieve the selected events one by one (The Selected property indicates whether the event is selected or unselected). We recommend using the Selection property and the for each statement to enumerate the events in the control. *The /NET and /WPF versions of the component provide the SelEvents function that retrieves a collection of Event objects, as List<Event>.* The Selectable property of the Event indicates whether the event can be selected at runtime.

Once the user starts selecting a new event in the schedule panel, the control fires the <u>LayoutStartChanging</u>(exScheduleSelectionChange). Once a new event is selected, the <u>LayoutEndChanging</u>(exScheduleSelectionChange) event occurs. You can use the <u>AllowSelectEvent</u> property to change the key to allow the user select new events or you can prevent selecting any event using exDisallow value.

method Schedule.SelectAll ()

Selects all events in the control.

Туре

Description

property Schedule.SelectEventColor as Color

Indicates the color to show the selected events.

Туре	Description
Color	A Color expression that defines color to show the frame around the selected event, or the background color of the event's body

If the <u>SelectEventStyle</u> property is exNoLines (by default), the SelectEventColor / <u>SelectEventTextColor</u> indicates the background / foreground colors to be applied on the event's body. If the SelectEventStyle property is NOT exNoLines, the the SelectEventColor property indicates the color to show the frame around the selected events. *In other words, you can use the SelectEventStyle on exLinesSolid* + *exLinesThick, to display a frame arround the selected events, rather than changing the event's background/foreground colors.* The <u>ShowSelectEvent</u> property prevents showing the selected events in a different way (using a frame around, by changing the body's background/foreground colors).

You can use the <u>AllowSelectEvent</u> property to change the key to allow the user select new events or you can prevent selecting any event using exDisallow value.

property Schedule.SelectEventStyle as LinesStyleEnum

Specifies the style to display the selected event.

Туре	Description
LinesStyleEnum	A LinesStyleEnum expression that defines the borders to be shown around the selected event.

By default, the SelectEventStyle property is exNoLines. If the SelectEventStyle property is exNoLines (by default), the <u>SelectEventColor</u> / <u>SelectEventTextColor</u> indicates the background / foreground colors to be applied on the event's body. If the SelectEventStyle property is NOT exNoLines, the the <u>SelectEventColor</u> property indicates the color to show the frame around the selected events. *In other words, you can use the SelectEventStyle on exLinesSolid* + *exLinesThick, to display a frame arround the selected events, rather than changing the event's background/foreground colors.* The <u>ShowSelectEvent</u> property prevents showing the selected events in a different way.

You can use the <u>AllowSelectEvent</u> property to change the key to allow the user select new events or you can prevent selecting any event using exDisallow value.

property Schedule.SelectEventTextColor as Color

Indicates the color to show the text for selected events.

Туре	Description
Color	A Color expression that specifies the selected event's foreground color.

The SelectEventTextColor property indicates the color to show the text/label/captions for the selected events. The SelectEventTextColor property has effect only, if the <u>SelectEventStyle</u> property is exNoLines (by default). SelectEventTextColor property has NO effect, if the the <u>SelectEventStyle</u> property is different than exNoLines value. The <u>ShowSelectEvent</u> property prevents showing the selected events in a different way (using a frame around, by changing the body's background/foreground colors).

You can use the <u>AllowSelectEvent</u> property to change the key to allow the user select new events or you can prevent selecting any event using exDisallow value.

property Schedule.Selection as Variant

Returns or sets a safe array of selected events in the schedule panel.

Туре	Description
Variant	A safe array of <u>Event</u> objects being or to be selected.

The Selection property of control can be used to set or get the current selection (events) in the control's schedule panel. The <u>Selection</u> property of the Calendar object can be used to set or get the current selection (dates) in the control's calendar panel. *The /NET and /WPF versions of the component provide the SelEvents function that retrieves a collection of Event objects, as List<Event>.* The <u>Selectable</u> property of the Event indicates whether the event can be selected at runtime. The <u>Selected</u> property indicates whether the event is selected or unselected.

Once the user starts selecting a new event in the schedule panel, the control fires the LayoutStartChanging(exScheduleSelectionChange). Once a new event is selected, the LayoutEndChanging(exScheduleSelectionChange) event occurs. You can use the <u>AllowSelectEvent</u> property to change the key to allow the user select new events or you can prevent selecting any event using exDisallow value.

The following sample shows how you can enumerate the selected events, once the LayoutEndChanging(exScheduleSelectionChange) event occurs.

VB

```
Private Sub Schedule1_LayoutEndChanging(ByVal Operation As
EXSCHEDULELibCtl.LayoutChangingEnum)
If Operation = exScheduleSelectionChange Then
Dim d As Variant
For Each d In Schedule1.Selection
Debug.Print "Event: " & d.Start & " " & d.End
Next
End If
End Sub
```

VB/NET

Private Sub Exschedule1_LayoutEndChanging(ByVal sender As System.Object, ByVal Operation As exontrol.EXSCHEDULELib.LayoutChangingEnum) Handles Exschedule1.LayoutEndChanging If Operation =

```
exontrol.EXSCHEDULELib.LayoutChangingEnum.exScheduleSelectionChange Then
Dim evs As List(Of exontrol.EXSCHEDULELib.Event) = Exschedule1.SelEvents
If Not evs Is Nothing Then
For Each d As exontrol.EXSCHEDULELib.Event In evs
Debug.Print("Event: " & d.Start & " " & d.End)
Next
End If
End If
End If
End Sub
```

C#

```
private void exschedule1_LayoutEndChanging(object sender,
exontrol.EXSCHEDULELib.LayoutChangingEnum Operation)
{
    if ( Operation ==
    exontrol.EXSCHEDULELib.LayoutChangingEnum.exScheduleSelectionChange )
        {
        List<exontrol.EXSCHEDULELib.Event> evs = exschedule1.SelEvents;
        if ( evs != null )
            foreach (exontrol.EXSCHEDULELib.Event d in evs)
            System.Diagnostics.Debug.Print("Event: " + d.Start.ToString() + " " +
d.Start.ToString());
        }
}
```

VFP

```
*** ActiveX Control Event ***
LPARAMETERS operation
* 10 ' exScheduleSelectionChange
If Operation = 10 Then
local e as Object
For Each e In thisform.schedule1.Selection
LOCAL ee as Object
ee = thisform.Schedule1.Events(e)
WAIT WINDOW TTOC(ee.Start) + " " + TTOC(ee.End)
```

```
ENDFOR
EndIf
```

C++

```
void CAddEventsDlg::LayoutEndChangingSchedule1(long Operation)
{
  if (Operation = = EXSCHEDULELib::exScheduleSelectionChange)
  {
    _variant_t evs = m_spSchedule->Selection;
    if (V_VT( &evs) == (VT_ARRAY | VT_VARIANT ))
    {
      BYTE* p = NULL;
      long nCount = 0;
      if (SUCCEEDED(SafeArrayGetUBound(V_ARRAY(&evs), 1, &nCount)))
      {
        if ( SUCCEEDED( SafeArrayAccessData( V_ARRAY( &evs ), (LPVOID*)&p ) ) )
        {
          for (long i = 0; i < nCount + 1; i++, p += sizeof(VARIANT))
           {
             VARIANT* pValue = (VARIANT*)p;
             if (V_VT(pValue) = VT_DISPATCH)
             {
               EXSCHEDULELib::IEventPtr spEvent = V_DISPATCH( pValue );
               CString strMessage;
               strMessage.Format(_T("Event: %f %f\r\n"), spEvent->Start, spEvent->End
);
               OutputDebugString( strMessage );
             }
          SafeArrayUnaccessData( V_ARRAY( & evs ) );
        }
      }
    }
```

where m_spSchedule is of EXSCHEDULELib::ISchedulePtr type.

property Schedule.SelEvent (Index as Long) as Event

Gets the event being selected giving its index in the selection.

Туре	Description
Index as Long	A Long expression that defines the index of the Event to be requested
<u>Event</u>	The Event being requested, if the index is between 0 and SelCount - 1, or empty/nothing/NULL

The <u>SelCount</u>/SelEvent property may be used to retrieve the selected events one by one (The <u>Selected</u> property indicates whether the event is selected or unselected). We recommend using the <u>Selection</u> property and the for each statement to enumerate the events in the control. *The /NET and /WPF versions of the component provide the SelEvents function that retrieves a collection of Event objects, as List<Event>.* The <u>EventFromPoint</u> property gets the event from the cursor. The <u>Selectable</u> property of the Event indicates whether the event can be selected at runtime.

You can use the <u>AllowSelectEvent</u> property to change the key to allow the user select new events or you can prevent selecting any event using exDisallow value. Once the user starts selecting a new event in the schedule panel, the control fires the <u>LayoutStartChanging</u>(exScheduleSelectionChange). Once a new event is selected, the <u>LayoutEndChanging</u>(exScheduleSelectionChange) event occurs.

property Schedule.ShowAllDayHeader as Boolean

Specifies whether the control shows or hides the header for All-Day events.

Туре	Description
Boolean	A Boolean expression that specifies whether the control's All-Day header is displayed or hidden.

By default, the ShowAllDayHeader property is False. Use the ShowAllDayHeader property to show the All-Day header, so all All-Day events are displayed on this header instead in the time scale section of the schedule view. The <u>AllDayEvent</u> property indicates whether the event is an All-Day event. Clicking the All-Day header makes the control to add a new All-Day event when user drops the mouse to a new position. The <u>AllowUpdateAllDayFlag</u> property specifies whether the event's <u>AllDayEvent</u> property is changed when the user drags an event from All-Day header to Time-Scale or reverse. The <u>AllowUpdateAllDayFlag</u> property on False indicates that the user can not drag an All-Day event to Time-Scale and reverse. The <u>HeaderAllDayEventHeight</u> property specifies the height of the events to be displayed in the All-Day header. The <u>AllowAllDayEventScroll</u> property gets or sets a value that specifies whether the all-day event header supports scrolling.

The following screen shot shows the All-Day events on the All-Day header (ShowAllDayHeader property is True):

	Sunday, May20, 1	2 Monday, May21, 12	Tuesday, May22, '12	Wednesday, May23, 12	Thursday, May24, '12	Friday, May 25, 12	Saturday, May26, 12
. 4	6/9/2012 - 6/31/2012						,
(C)	4 5/18/2012 - 5/23/2012				6/24/2012 - 5/25/2012 All-Day E	vent	M-Z
Õ	5/20/2012 - 5/23/2012				5/24/2012 - 5/25/2012 All-Dwy E	vent K-M	
_	and the second state of the second		5/22/2012 - 5/23/2012			5/25/2012	
∢_				5/23/2012 - 5/24/2012 All-Day	Event A-K	5/25/2012 - 5/28/2012	
	08:00 AM			8:00 AM - 10:00 AM			
•	:20 :40 8:40 AM - 12:40 PM						
	09:00 AM .20	9 00 AM - 11:40 AM	9:00 AM - 1:20 PM				9.10 AM - 2.30 PM
0.0	:40 10:00 AM				10:00 AM - 12:00 PM	10.10 AM	
ale	20 40					10.10 Mill - 12.00 Fill	
Sci	11:00 AM :20			11-20 AM - 2:10 PM			
d	:40 12:00 PM				dentist	12.40 PM 2.00 PM	-
Ĕ	20	12:20 PM - 2:30 PM				12.00 PM - 2.00 PM	
F	01:00 PM		-		1:00 PM - 3:10 PM		
	:20						
	02:00 PM						
1							
	03:00 PM				+		

The following screen shot shows the All-Day events on the All-Day header (ShowAllDayHeader property is False):

		Su	nday,M	tay20,	12	M	onday.	May21	1,12		Tuesd	ay.Ma	ay 22,	12	N	ednes	day.N	lay23	12	Ţ	hurse	lay, Ma	y24,	12		Frida	iy. May	25.	12	Sat	turday.	May20	6,12
06.00	AU .20	5/9V	517	5/2		5/9/.	5/1	5/2		5/	5/	5/ -	51		5	5	5 5		8 AM	s	sv	6/	s/		5	5	5		5	5/2	5/9/	5/2	
1,000	40	5/3	5/2	0/Z	8:40	5/3	5/2	5/2		5/	5/	o/	5/		0	5	5 5	5	1	5/ AL	s/	AL_	5/		5	5	5			5/2	5/3	5/2	
09:00	AN				AM - 12:40			1	9:00 AM		L	L		9:00	E.	11		Т	AM	E	Al	E			E.,	E.		1		Event			9:
2	40				PM			L .	11:40		L	L		-		11		н			-							1					A 2
10.00	AU	5 8							~		۰.			PM		11				-	Ł			1				1					P
	20							L .			L	L				11		Т						AM -				1	1. AM			L	Т
>	-40							L .			L	L				11		L			1			1 PM				1	1				L
11:00	-20							L .			L	L				11		Т			L .							1	PM			L 1	I
	:40							L .			L	L				11		Е	AM		L							1					I
12.00	PU							L .			t.	L				11		1	2		1					11		1	1				I
	:20							L .	12:20 PM -		L	L				11		н	PM									1	-			L	I
01-00	40								2:30 PM	-	۰.	L				11					+			1.00				1	2 PM				ł
01.00	20							L .			L	L				11		Т						PM				1					L
	:40							L .			L	L				11		н			L .			3:10				1					L
02.00	PM															11		1	•	1	1			PM		11		1					1
	40																																
03:00	PM	1.13				-										11																	
1	20						1																										
1	:40	-	•	•		-	1.	-			-	1.	-										-						-		-		

property Schedule.ShowEventLabels as Boolean

Indicates whether the Label or ExtraLabel of the events are being shown or hidden.

Туре	Description
Boolean	A boolean expression that specifies whether the labels of the events are shown or hidden.

By default, the ShowEventLabels property is True. The ShowEventLabels property specifies whether the labels are shown on the event's body. The <u>ShortLabel</u>, <u>LongLabel</u> or <u>ExtraLabel</u> property indicates labels that could be displayed on any event. The <u>ShowEvents</u> property specifies what events the control should show. The <u>ShowEventPictures</u> property specifies whether the labels are shown on the event's body. The <u>EventsTransparent</u> property indicates the transparency to show the events on the schedule view.

property Schedule.ShowEventPictures as Boolean

Indicates whether the Pictures or ExtraPictures of the events are being shown or hidden.

Туре	Description
Boolean	A boolean expression that specifies whether the pictures of the events are shown or hidden.

By default, the ShowEventPictures property is True. The ShowEventPictures property specifies whether the labels are shown on the event's body. The <u>ShowEventLabels</u> property specifies whether the labels are shown on the event's body. The <u>Pictures</u>, or <u>ExtraPictures</u> property indicates pictures to be shown on the event's body. The <u>ShowEvents</u> property specifies what events the control should show. The <u>EventsTransparent</u> property indicates the transparency to show the events on the schedule view. The <u>Clear</u> method of the ExPictures collection clears the control's pictures collection.

property Schedule.ShowEvents as ShowEventsEnum

Indicates the type of the events which schedule displays.

Туре	Description
<u>ShowEventsEnum</u>	A ShowEventsEnum expression indicates the type of events the control shows.

By default, the ShowEvents property is exShowAllEvents. The ShowEvents property specifies what events the control should show. You can use the ShowEvents property to show only the regular events, repetitive events, or all events. The <u>ShowEventLabels</u> property specifies whether the labels are shown on the event's body. The <u>ShowEventPictures</u> property specifies whether the labels are shown on the event's body. The <u>ShowEventPictures</u> property specifies whether the labels are shown on the event's body. The <u>ShowEventPictures</u> property indicates the transparency to show the events on the schedule view.

For instance, the ShowEvents on 0 (zero), indicates no events are shown on the control. the ShowEvents on 2 (two), indicates that the schedule view displays the repetitive events only.
property Schedule.ShowGroupingEvents as Boolean

Specifies whether the schedule view shows grouped events.

Туре	Description
Boolean	A Boolean expression that specifies whether the control displays events based on the owner groups.

By default, the ShowGroupingEvents property is False. The ShowGroupingEvents property indicates whether the control displays events grouped by its <u>GroupID</u> property. The <u>DisplayGroupingButton</u> property indicates whether the header of the date displays the grouping button. The list of available groups is displayed on a drop down panel, once the user clicks the grouping/filtering button. The drop down list shows the <u>Title</u> for each group found. The <u>ApplyGroupingColors</u> property specifies whether the control uses the Group's <u>EventBackColor</u> / <u>EventForeColor</u> / <u>EventPattern</u> properties to show the events in the groups. The <u>SingleGroupingView</u> property specifies whether the drop down panel shows radio buttons, instead check boxes, so the user can see all groups or only one group at the time. The <u>GroupID</u> property specifies the identifier of the group where the event belongs. If the control displays groups the GroupID property of the newly created event is automatically updated with the group where the event has been created. The <u>AllowMoveEventToOtherGroup</u> property specifies whether the user can move an event from a group to another at runtime. Use the <u>Add</u> method of the Groups collection to add new groups to the control.

The grouping button is displayed if:

- DisplayGroupingButton property is True
- ShowGroupingEvents property is True

The control displays groups if:

- ShowGroupingEvents property is True
- The <u>Groups</u> collection has elements. By default, the Groups collection contains no Group objects.

The following <u>Background</u> properties change the visual appearance of the drop down grouping panel:

- Background(exGroupingBackColor) / Background(exGroupingForeColor) changes the background and the foreground color of the panel.
- Background(exGroupingSelBackColor) / Background(exGroupingSelForeColor) changes the background and the foreground color of the selection in the panel.
- Background(exCheckBoxState0), Background(exCheckBoxState1), Background(exCheckBoxState2) changes the visual appearance for the control's check

boxes.

• Background(exRadioButtonState0), Background(exRadioButtonState1), changes the visual appearance for the control's radio buttons.

The <u>Description</u>(exGroupBarAll) property changes the "(All)" predefined string, being displayed on the top of the drop down grouping/filtering panel.

The following screen shot shows the drop down panel, if the <u>SingleGroupingView</u> property is False (by default):



The following screen shot shows the drop down panel, if the <u>SingleGroupingView</u> property is True:



property Schedule.ShowHighlightDate as ShowHighlightDateEnum

Returns or sets a value that indicates whether the control shows the highlighted dates.

Туре

Description

<u>ShowHighlightDateEnum</u>

A <u>ShowHighlightDateEnum</u> expression that specifies the way the control shows highlighted dates.

By default, the ShowHighlightDate property is exShowHighlightDate, which means that the date is highlighted in the calendar and schedule panels. The ShowHighlightDate property can highlight the date in the calendar panel and the header of the date in the schedule panel. Use the ShowHighlightDate property to specify whether the highlighted dates are shown on calendar or/and schedule panels. The <u>HighlightDate</u> property highlights/un-highlights the specified date with the giving color(s). The <u>HighlightEvent</u> property highlights dates in the calendar panel, when it contains events. The <u>ScheduleHighlightEvent</u> property specifies the visual appearance of dates with events in the schedule panel.

The following screen shot shows a few dates highlighted with different colors:

•		Jan	nuar	y 20	01		•							
							Sa	7	8	9	10	11	12	13
	31	1	2	3	4	5	6							
	7	8	0	10	11	12	13		45	40	47	40	10	20
	14	15	16	1	18	19	20	14	15	10	14	10	19	20
	21	22	23	24	0	26	27				-		19	
	28	29	30	31	1	2	3	21	22	23	24	25	26	27
	4	5	6	7	8	9	10							
								28	29	30	31			
		ſ	Too	lay)		_							

property Schedule.ShowHighlightEvent as Boolean

Returns or sets a value that indicates whether the schedule panel highlights days that contain events.

Туре	Description
Boolean	A Boolean expression that specifies whether the schedule panel highlights the dates with events or appointments

Use the <u>ShowHighlightEvent</u> property of the Calendar object, to customize the dates with events, in the calendar panel.

By default, the ShowHighlightEvent property is true, which indicates that the dates with events or appointments appear as bold in the schedule panel. You can use the ShowHighlightEvent property to prevent highlighting the the dates with events or appointments. You can use the <u>HighlightEvent</u> object to highlight the dates with events or appointments in the calendar panel. The <u>GroupHighlightEvent</u> property specifies if events are highlighted using the HighlightEvent property (False), or using the <u>CalendarHighlightEvent</u> property of the <u>Group</u> that event belongs to (True). The <u>ScheduleHighlightEvent</u> property specifies the visual appearance of dates with events in the schedule panel.

Using the Highlight object a date with events can combine one or more of the following options:

- **bold**, **Bold** property renders as bold text
- *italic*, *Italic* property renders as italic text
- <u>underline</u>, <u>Underline</u> property underlines the text
- strikeout, StrikeOut property shows the text with a horizontal line through its center
- change the font SiZe, <u>FontSize</u> property indicates the size of the font to display the text
- change the **font**, using the <u>Font</u> property
- change the text's foreground color, using the ForeColor property
- change the text's background color, using the BackColor property
- shows a pattern using the Pattern property

property Schedule.ShowImageList as Boolean

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

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

The property is available for /COM version only, and only at design mode. By default, the ShowImageList property is False. Use the ShowImageList property to show the control's images list window. The control's images list window is visible only at design time. Use the <u>Images</u> method to associate an images list control to the control. Use the <u>ReplaceIcon</u> method to add, remove or clear icons in the control's images collection, at runtime.

property Schedule.ShowMarkTime as Boolean

Indicates whether the schedule shows the mark times.

Туре	Description
Boolean	A Boolean expression that specifies whether the control shows or hides the timers.

By default, The ShowMarkTime property is True, which indicates that all added timers are visible. The ShowMarkTime property indicates whether the schedule view displays timers. The <u>AllowMoveMarkTime</u> property indicates the keys to allow user to move timers (with the <u>Movable</u> property on True). The <u>MarkTimeFromPoint</u> property indicates the timer from the cursor. The <u>MarkTimes</u> property gets a collection of MarkTime objects, also called timers. The <u>MarkTime</u> object indicates a line in the schedule view, at a specified time. The <u>Add</u> method of <u>MarkTimes</u> collection adds a new timer to the schedule view. The MarkTimes collection is accessible through the MarkTimes property of the control.

The MarkTime object, also called **timer**, can be used to:

- show a line of different styles on the schedule view, at specified time
- show a HTML label at specified time
- highlights the events that intersect with the timer

property Schedule.ShowMarkZone as ShowMarkZoneEnum

Indicates how the schedule panel shows the mark zones.

_	_			
		1.1	6	r n
	. 1	10		
		100	24	~~

Description

<u>ShowMarkZoneEnum</u>

A ShowMarkZoneEnum expression that specifies how the time-zones are shown on the control.

The ShowMarkZone property shows or hides the added time-zones. Using the ShowMarkZone property the mark zones can be shown:

- hidden, exHideMarkZones
- on the back of the other elements as events, and so on, exShowMarkZonesBack
- on the front of the other elements as events, and so on, exShowMarkZonesFront (by default)
- using a semi-transparent color, exShowMarkZonesSemi (by default)

The <u>MarkZones</u> property gets the MarkZones collection. The MarkZones collection holds a set of <u>MarkZone</u> objects (also called time-zone). A MarkZone object holds information about a time-zone. A time-zone is identified by a <u>Start/End</u> date time, what can be highlighted in the schedule view. Use the <u>Add</u> method of the MarkZones collection to add a new time-zone to the control. The <u>MarkZoneFromPoint</u> property indicates the time-zone from the cursor.

The MarkZone object can:

- show a HTML/Image caption on a specified time-zone
- highlight a time-zone with a different background, pattern and so on, to indicate a restricted zone for instance.

A time-zone (<u>MarkZone</u> object) requires the <u>Start/End</u> to define the zone, while a timer (<u>MarkTime</u> object) requires a <u>Time</u>, that indicates where the timer is shown.

The following screen shot shows the time-zones on the front, exShowMarkZonesFront (by default):



The following screen shot shows the time-zones on the back, exShowMarkZonesBack:



The following screen shot shows the time-zones using a semi-transparent color, exShowMarkZonesSemi:



property Schedule.ShowNonworkingTime as ShowNonworkingTimeEnum

Returns or sets a value that indicates whether the schedule panel displays nonworking time.

Туре

Description

ShowNonworkingTimeEnum

A ShowNonworkingTimeEnum expression that defines how the non-working time-zones are displayed.

By default, the ShowNonworkingTime property is exShowNonworkingTimeFront. The ShowNonworkingTime property shows or hides the defined non-working intervals. The Add method of the NonworkingTimes objects adds a new non-working time interval. The NonworkingTimes collection is accessible through the NonworkingTimes property of the control. The Expression property indicates the expression that defines the dates to include the specified non-working interval. The IsValid property indicates whether the non-working expression is valid, and so, if it is visible or hidden. The StartTime/EndTime property defines the time to start/end the non-working time-zone. The NonworkingTimeFromPoint property gets the non-working object from the cursor.

The NonworkingTimes object holds a collection of <u>NonworkingTime</u> objects. The NonworkingTime object indicates a time interval to be shown as non-working. Each NonworkingTime object can associate a <u>NonworkingPattern</u> object that specifies the colors and the pattern to show the non-working zone. The <u>NonworkingPatterns</u> collection is accessible through the <u>NonworkingPatterns</u> property of the control. The <u>NonworkingDays</u> property of the calendar defines the days to be non-working in the calendar.

The NonworkingTime's advantages are:

- highlight the interval of time as non-working with a different patterns, colors.
- any/all day can display different intervals of time as non-working
- you can specify the non-working interval using an expression, that defines the days where the non-working interval is shown.

property Schedule.ShowSelectEvent as Boolean

Specifies whether the selected events are highlighted.

Туре	Description
Boolean	A boolean expression that specifies whether the selected events are shown in a in a different way,

By default, the ShowSelectEvent property is True. The ShowSelectEvent property prevents showing the selected events in a different way (using a frame around, by changing the body's background/foreground colors). If the <u>SelectEventStyle</u> property is exNoLines (by default), the <u>SelectEventColor</u> / <u>SelectEventTextColor</u> indicates the background / foreground colors to be applied on the event's body. If the SelectEventStyle property is NOT exNoLines, the the <u>SelectEventColor</u> property indicates the color to show the frame around the selected events. *In other words, you can use the SelectEventStyle on exLinesSolid* + *exLinesThick, to display a frame arround the selected events, rather than changing the event's background/foreground colors.*

You can use the <u>AllowSelectEvent</u> property to change the key to allow the user select new events or you can prevent selecting any event using exDisallow value.

property Schedule.ShowStatusEvent as Boolean

Gets or sets a value that specifies whether the event's status is visible or hidden.

Туре	Description
Boolean	A Boolean expression that specifies whether the status part of the event is shown or hidden.

By default, the ShowStatusEvent property is True, which means that the control shows the status part of the event. The ShowStatusEvent property shows or hides the status part for all events. The <u>ShowStatus</u> property shows or hides the status part of giving event. Use the <u>ClearShowStatus</u> method to allow the ShowStatusEvent property to display the event's status, rather than <u>ShowStatus</u> property. The <u>StatusEventColor</u> property indicates the color to show the status part of the events. The <u>StatusEventSize</u> property specify the size in pixels of the event's status.

You can:

- show the status part for all events (ShowStatusEvent on True), and use the ShowStatus (<u>ShowStatus</u> on False) property to hide the status part for specified events only.
- hide the status part for all events (ShowStatusEvent on False), and use the ShowStatus (<u>ShowStatus</u> on True) property to show the status part for specified events only.

property Schedule.ShowTimeScale as Boolean

Specifies whether the control's time scale is shown on the schedule panel.

Туре	Description
Boolean	A boolean expression that indicates whether the control display the time-scale.

The control displays time scale if it fits on the control's client area. The control decides when it can display the time-scale depending on how large is the schedule view. The <u>Visible</u> property of the <u>TimeScale</u> object indicates whether the control displays or hides the time-scale.

The following screen shot shows the control with no time-scale (ShowTimeScale property is False):



The following screen shot shows the control with the default time-scale (ShowTimeScale property is True):



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

Shows the specified tooltip at given position.

Туре	Description
	The ToolTip parameter can be any of the following:
ToolTip as String	 NULL(BSTR) or "<null>"(string) to indicate that the tooltip for the object being hovered is not changed</null> 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.</null> A String expression that indicates the title of the tooltip (no built-in HTML format) (adds, replaces or changes the object's title)
	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:
Alignment as Variant	 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).

	Specifies the horizontal position to display the tooltip as one of the following:
X as Variant	 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)
	Specifies the vertical position to display the tooltip as one of the following:
Y as Variant	 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 <u>MouseMove</u> event. Use the <u>ToolTipPopDelay</u> property specifies the period in ms of time the ToolTip remains visible if the mouse pointer is stationary within a control. The <u>ToolTipDelay</u> property specifies the time in ms that passes before the ToolTip appears. Use the <u>ToolTipWidth</u> property to specify the width of the tooltip window. Use the <u>ToolTipFont</u> property to change the tooltip's font. Use the <u>Background(exToolTipAppearance)</u> property indicates the visual appearance of the borders of the tooltips. Use the <u>Background(exToolTipBackColor)</u> property indicates the tooltip's foreground color. Use the <u>Background(exToolTipForeColor)</u> 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**
- <i> ... </i> displays the text in *italics*
- <u> ... </u> <u>underlines</u> the text
- <s> ... </s> Strike-through text
- <a id;options> ... displays an <u>anchor</u> element that can be clicked. An anchor is a piece of text or some other object (for example an image) which marks the beginning and/or the end of a hypertext link. The <a> element is used to mark that piece of text (or inline image), and to give its hypertextual relationship to other documents. The control fires the AnchorClick(AnchorID, Options) event when the user clicks the anchor element. The FormatAnchor property customizes the visual effect for anchor elements.

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

- exp, stores the plain text to be shown once the user clicks the anchor, such as "<a ;exp=show lines>"
- $\circ~$ e64, encodes in BASE64 the HTML text to be shown once the user clicks the anchor, such as "<a $\!$

;e64=gA8ABmABnABjABvABshIAOQAEAAHAAGESikWio+ABzABohp3iELABpABu " that displays show lines- in gray when the user clicks the + anchor. The "gA8ABmABnABjABvABshIAOQAEAAHAAGESikWio+ABzABohp3iELABpABuABljY string encodes the "<fgcolor 808080>show lines<a>-</fgcolor>" The Decode64Text/Encode64Text methods of the eXPrint can be used to decode/encode e64 fields.

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

- ... displays portions of text with a different font and/or different size. For instance, the "bit" draws the bit text using the Tahoma font, on size 12 pt. If the name of the font is missing, and instead size is present, the current font is used with a different size. For instance, "bit" displays the bit text using the current font, but with a different size.
- **<fgcolor rrggbb>** ... **</fgcolor>** or **<**fgcolor=rrggbb> ... **</**fgcolor> displays text with a specified foreground color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- <bgcolor rrggbb> ... </bgcolor> or <bgcolor=rrggbb> ... </bgcolor> displays text
 with a specified background color. The rr/gg/bb represents the red/green/blue values of
 the color in hexa values.
- <solidline rrggbb> ... </solidline> or <solidline=rrggbb> ... </solidline> draws a solidline 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
- number[:width] 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.
- key[:width] 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 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 FFFFF;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 Schedule.ShowViewCompact as ShowViewCompactEnum

Indicates whether the schedule view is compact, so the first day of the month starts right after the last day of the previously month, or start to a new row.

Туре

Description

ShowViewCompactEnum

A ShowViewCompactEnum expression that specifies whether the schedule view is shown compact.

By default, the ShowViewCompact property is exViewCalendar(0). The ShowViewCompact property specifies the way the control shows/layouts the dates in the schedule view.

The ShowViewCompact property can arrange the dates, in the schedule view:

- exViewCalendar (0), as they are shown in the calendar panel
- **exViewCalendarCompact** (-1), as they are shown in the calendar panel, excepts that the first day of the month starts right after the last day of the previously month, or start to a new row. This option is valid, ONLY, if the calendar panel displays 1 x 12 month. In other words, the Calendar.<u>MinMonthX</u> and <u>MaxMonthX</u> properties must be set on 1 (by default).
- **exViewSingleRow** (1), in a single row only, so the view displays days one after other, no matter of what dates are selected in the calendar panel.

The following screen shot shows the schedule view, for ShowViewCompact property on **exViewCalendar**:

									Sunday, 20	Monday, 21	Tuesday, 22	Wednesday, 23	Thursday, 24	Friday, 25	Saturday, 26
								10:00 AM							
								11:00 AM							
								:30							
								:30							
			May	2012	2		•	01:00 PM 30							
	Su	Мо	Tu	We	Th	Fr	Sa		Sunday 27	Monday 28	Tuesday 29	Wednesday 30	Thursday 31		
8	29	30	1	2	3	4	5	10:00 AM	ounday, 21	monday, 20	10000037, 20	Troundoday, oo	maroday, or		
9	6	7	8	9	10	11	12	:30							
0	13	14	15	16	17	18	19	11:00 AM :30							
1		21	22	23	24	25		12:00 PM							
2		28	29	30	31	1		01:00 PM							
3	3	4	5	6	7	8	9	:30							
														Friday, 1	Saturday, 2
								10:00 AM						*****	***************************************
								11:00 AM							
								:30							
								:30							
-	_	_	Te	day	1		-	01:00 PM							
			10	uay	J			:30							

The following screen shot shows the schedule view, for ShowViewCompact property on **exViewCalendarCompact**:

							1		Sunday, 20	Monday, 21	Tuesday, 22	Wednesday, 23	Thursday, 24	Friday, 25	Saturday, 26
								10:00 AM :15 :30 :45							
•			May	2012	2		•	11:00 AM :15 :30 :45 12:00 PM							
	Su	Мо	Tu	We	Th	Fr	Sa	:30							
18	29	30	1	2	3	4	5	01:00 PM							
19	6	7	8	9	10	11	12	:15 :30 :45							
20	13	14	15	10	24	25	19		Sunday, 27	Monday, 28	Tuesday, 29	Wednesday, 30	Thursday, 31	Friday, 1	Saturday, 2
22		28	29	30	31	1		10:00 AM							
23	3	4	5	6	7	8	9	:30							
	67.0010							11:00 AM :15 :30 :45							
								12:00 PM :15 :30 :45							
			То	day)		_	01:00 PM :15 :30 :45							

The following screen shot shows the schedule view, for ShowViewCompact property on **exViewSingleRow**:

									Sun	Mon	Tue	Wed	Thu	Fri	Sat	Sun	Mon	Tue	Wed	Thu	Fri	Sat
								10:00 AM														
								:15														
								:30														
								:45														
•			May	2012	2		•	11:00 AM														
	Su	Мо	Tu	We	Th	Fr	Sa	-15														
18	29	30	1	2	3	4	5	-20														
19	6	7	8	9	10	11	12	.50														
20	13	14	15	16	17	18	19	:45														
21	24	21	22	23	24	25	25	12:00 PM														
22	21	28	29	30	31	1		:15														
23	3	4	5	6	7	8	9	:30														
								:45														
								01:00 PM														
								:15														
								:30														
								-45														
_			То	dav	1			.40														
				-	,			(4)					Cu	0								01

property Schedule.SingleGroupingView as Boolean

Indicates whether the schedule shows single or multiple groups of events at once.

Туре	Description
Boolean	A Boolean expression that specifies whether the drop down grouping list displays radio buttons rather than check boxes.

By default, the SingleGroupingView property is False. The SingleGroupingView property specifies whether the drop down panel shows radio buttons, instead check boxes, so the user can see all groups or only one group at the time. The <u>DisplayGroupingButton</u> property indicates whether the header of the date displays the grouping button. The list of available groups is displayed on a drop down panel, once the user clicks the grouping/filtering button. The drop down list shows the <u>Title</u> for each group found. The <u>ShowGroupingEvents</u> property indicates whether the control displays events grouped by its <u>GroupID</u> property. The <u>ApplyGroupingColors</u> property specifies whether the control uses the Group's <u>EventBackColor</u> / <u>EventForeColor</u> / <u>EventPattern</u> properties to show the events in the groups. The <u>GroupID</u> property specifies the identifier of the group where the event belongs. If the control displays groups the GroupID property of the newly created event is automatically updated with the group where the event has been created. The <u>AllowMoveEventToOtherGroup</u> property specifies whether the user can move an event from a group to another at runtime. Use the <u>Add</u> method of the Groups collection to add new groups to the control.

The grouping button is displayed if:

- DisplayGroupingButton property is True
- ShowGroupingEvents property is True

The control displays groups if:

- ShowGroupingEvents property is True
- The <u>Groups</u> collection has elements. By default, the Groups collection contains no Group objects.

The following <u>Background</u> properties change the visual appearance of the drop down grouping panel:

- Background(exGroupingBackColor) / Background(exGroupingForeColor) changes the background and the foreground color of the panel.
- Background(exGroupingSelBackColor) / Background(exGroupingSelForeColor) changes the background and the foreground color of the selection in the panel.
- Background(exCheckBoxState0), Background(exCheckBoxState1),

Background(exCheckBoxState2) changes the visual appearance for the control's check boxes.

• Background(exRadioButtonState0), Background(exRadioButtonState1), changes the visual appearance for the control's radio buttons.

The following screen shot shows the drop down panel, if the <u>SingleGroupingView</u> property is False (by default):



The following screen shot shows the drop down panel, if the <u>SingleGroupingView</u> property is True:



method Schedule.StartBlockUndoRedo ()

Starts recording the UI operations as a block of undo/redo operations.

Туре

Description

You can use the StartBlockUndoRedo / EndBlockUndoRedo methods to group multiple Undo/Redo operations into a single-block. The GroupUndoRedoActions groups the next to current Undo/Redo Actions in a single block. A block may hold multiple Undo/Redo actions. The AllowUndoRedo property enables or disables the Undo/Redo feature. Use the GroupUndoRedoActions method to group two or more entries in the Undo/Redo queue in a single block, so when a next Undo/Redo operation is performed, multiple actions may occur. For instance, moving several calendar-events in the same time (multiple calendar-events selection) is already recorded as a single block. Use the <u>UndoRedoQueueLength</u> property to specify the number of entries that Undo/Redo queue may store.

A block starts with StartBlock and ends with EndBlock when listed by <u>UndoListAction</u>/<u>RedoListAction</u> property as in the following sample:

StartBlock MoveEvent;B MoveEvent;A EndBlock

property Schedule.StatusEventColor as Color

Indicates the default visual appearance for the event's status.

Туре	Description
Color	A Color expression that specifies the visual appearance of the status part of the events. The last 7 bits in the high significant byte of the color to indicates the identifier of the skin being used. Use the <u>Add</u> 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.

By default, the StatusEventColor property is 0. The StatusEventColor property indicates the color to show the status part of the events. By default, the StatusEventColor property specifies the color to show the status part of the event. The <u>StatusColor</u> property indicates the color to show the status part of a specified event. The <u>StatusEventSize</u> property specify the size in pixels of the event's status.

You can:

- show the status part for all events (<u>ShowStatusEvent</u> on True), and use the ShowStatus (<u>ShowStatus</u> on False) property to hide the status part for specified events only.
- hide the status part for all events (<u>ShowStatusEvent</u> on False), and use the ShowStatus (<u>ShowStatus</u> on True) property to show the status part for specified events only.

property Schedule.StatusEventSize as Long

Indicates the size of the event's status.

Туре	Description
Long	A Long expression that defines the width of the status part of the event.

By default, the StatusEventSize property is 4 pixels wide. The StatusEventSize property specify the size in pixels of the event's status. The <u>ShowStatusEvent</u> property shows or hides the status part for all events. The <u>ShowStatus</u> property shows or hides the status part of giving event. Use the <u>ClearShowStatus</u> method to allow the ShowStatusEvent property to display the event's status, rather than <u>ShowStatus</u> property. The <u>StatusEventColor</u> property indicates the color to show the status part of the events.

You can:

- show the status part for all events (ShowStatusEvent on True), and use the ShowStatus (<u>ShowStatus</u> on False) property to hide the status part for specified events only.
- hide the status part for all events (ShowStatusEvent on False), and use the ShowStatus (<u>ShowStatus</u> on True) property to show the status part for specified events only.

method Schedule.Synchronize ([Records as Variant])

Synchronizes the control' events with the records, while the control is bounded to a recordset, using the DataSource property.

Туре	Description
Records as Variant	A VARIANT expression that specifies an empty expression or a safe array of bookmarks to be synchronized. If missing or empty, the entire control/recordset is synchronized

The Syncronize method updates the giving records from the control and back. For instance, the DAO provides no notification when a new record is added, so if you add new records by code, and want to have them added to the control, you need to use the Syncronize method. The Syncronize method has no effect if no <u>DataSource</u> property is used. In other words, the Syncronize method ensures that each record has associated an event, and each event has associated a record.

The following sample shows how you can syncronize the control events and the table once all records of the tabel has been deleted (MS Access, or DAO)

CurrentProject.Connection.Execute "DELETE FROM Events" Schedule1.Synchronize

Because DAO does not provide sync events you can use the Syncronize method to update the events of the control, once you update externally the table.

property Schedule.Template as String

Specifies the control's template.

Туре	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 <u>ExecuteTemplate</u> 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 Schedule.TemplateDef as Variant

Defines inside variables for the next Template/ExecuteTemplate call.

Туре	Description
Variant	A string expression that indicates the Dim declaration, or any Object expression to be assigned to previously declared variables.

The TemplateDef property has been added to allow programming languages such as dBASE Plus to set control's properties with multiple parameters. It is known that programming languages such as **dBASE Plus or XBasic from AlphaFive**, does not support setting a property with multiple parameters. In other words, these programming languages does not support something like *Property(Parameters) = Value*, so our controls provide an alternative using the TemplateDef method. The first call of the TemplateDef should be a declaration such as "Dim a,b" which means the next 2 calls of the TemplateDef defines the variables a and b. The next call should be <u>Template</u> or <u>ExecuteTemplate</u> 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
```

.ltems.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, <u>Template</u> and <u>ExecuteTemplate</u> 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 Schedule.TemplatePut (NewVal as Variant)

Defines inside variables for the next Template/ExecuteTemplate call.

Туре	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 / <u>TemplateDef</u> 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 <u>Template</u> or <u>ExecuteTemplate</u> property which can use the variable a and b being defined previously.

The <u>TemplateDef</u>, TemplatePut, <u>Template</u> and <u>ExecuteTemplate</u> 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 Schedule.TimeFromPoint (X as OLE_XPOS_PIXELS, Y as OLE_YPOS_PIXELS) as Date

Retrieves the time from the cursor, in the schedule panel.

Туре	Description
X as OLE_XPOS_PIXELS	A single that specifies the current X location of the mouse pointer. The x values is always expressed in client coordinates.
Y as OLE_YPOS_PIXELS	A single that specifies the current Y location of the mouse pointer. The y values is always expressed in client coordinates.
Date	A Date expression that indicates the time from the cursor. The value is between 0 and 1, and it indicates the time from the cursor.

The <u>MouseMove</u> event is generated continually as the mouse pointer moves across objects. During the MouseMove event you can call the <u>ShowToolTip</u> method to display any custom tooltip. During the <u>Click</u> or <u>RClick</u> event you can get an UI part of the control using one of the following properties. *All ...FromPoint properties can be use such as ...FromPoint(-1,-1) to get the UI part of the control from the current mouse position, in other words, you do not have to pass any X, Y coordinates.*

You can get UI parts from the cursor, using any of the following ... FromPoint properties:

- Calendar. <u>DateFromPoint</u> property gets the date from the cursor in the Calendar panel.
- <u>AnchorFromPoint</u> property gets the anchor (<a>) element from the cursor.
- <u>DateTimeFromPoint</u> property gets the date/time from the cursor, in the Schedule panel.
- <u>TimeFromPoint</u> property gets the time from the cursor, in the Schedule panel.
- EventFromPoint property gets the Event object from the cursor.
- <u>GroupFromPoint</u> property retrieves the <u>Group</u> object from the cursor. Use the GroupHeaderFromPoint property to the Group object when cursor hovers the group's header.
- <u>GroupHeaderFromPoint</u> property retrieves the <u>Group</u> object from the cursor, when the cursor hovers just the group's header.
- <u>MarkTimeFromPoint</u> property gets the <u>MarkTime</u> object from the cursor. A MarkTime object is identified by a date/time where it is displaying.
- <u>MarkZoneFromPoint</u> property gets the <u>MarkZone</u> object from the cursor. A MarkZone object is identified by a starting date/time and ending date/time where it is displaying.
- <u>NonworkingTimeFromPoint</u> property indicates the <u>NonworkingTime</u> object from the cursor.
- <u>PictureFromPoint</u> property indicates the key of the picture from the cursor.

• <u>TimeScaleFromPoint</u> property returns the <u>TimeScale</u> object from the cursor.

All ... FromPoint properties that returns an Object, may return Nothing, Empty or NULL, if no object is found.

property Schedule.TimeScaleFont as IFontDisp

Retrieves or sets the font to display the time scales in the schedule view.

Туре	Description
IFontDisp	A Font object used to paint the captions/labels in the time scale.

Use the TimeScaleFont property specifies the time scale's font. The ForeColor property indicates the foreground color to show the captions or labels.

The control supports the following Font properties:

- Font property, that specifies the control's font, including the Calendar's font
- <u>EventsFont</u> property, indicates the font to show the captions and labels on Event/Appointment objects
- TimeScaleFont property, specifies the font to display the labels on the control's time scales
- <u>ToolTipFont</u> property specifies the font to display the tooltip being shown when the cursor is hovering an part of the control.

property Schedule.TimeScaleFromPoint (X as OLE_XPOS_PIXELS, Y as OLE_YPOS_PIXELS) as TimeScale

Retrieves the TimeScale object from the cursor, in the schedule panel.

Туре	Description
X as OLE_XPOS_PIXELS	A single that specifies the current X location of the mouse pointer. The x values is always expressed in client coordinates.
Y as OLE_YPOS_PIXELS	A single that specifies the current Y location of the mouse pointer. The y values is always expressed in client coordinates.
<u>TimeScale</u>	A TimeScale object from the cursor.

The <u>MouseMove</u> event is generated continually as the mouse pointer moves across objects. During the MouseMove event you can call the <u>ShowToolTip</u> method to display any custom tooltip. During the <u>Click</u> or <u>RClick</u> event you can get an UI part of the control using one of the following properties. *All ...FromPoint properties can be use such as ...FromPoint(-1,-1) to get the UI part of the control from the current mouse position, in other words, you do not have to pass any X, Y coordinates.*

You can get UI parts from the cursor, using any of the following ... FromPoint properties:

- Calendar. <u>DateFromPoint</u> property gets the date from the cursor in the Calendar panel.
- AnchorFromPoint property gets the anchor (<a>) element from the cursor.
- <u>DateTimeFromPoint</u> property gets the date/time from the cursor, in the Schedule panel.
- <u>TimeFromPoint</u> property gets the time from the cursor, in the Schedule panel.
- EventFromPoint property gets the Event object from the cursor.
- <u>GroupFromPoint</u> property retrieves the <u>Group</u> object from the cursor. Use the GroupHeaderFromPoint property to the Group object when cursor hovers the group's header.
- <u>GroupHeaderFromPoint</u> property retrieves the <u>Group</u> object from the cursor, when the cursor hovers just the group's header.
- <u>MarkTimeFromPoint</u> property gets the <u>MarkTime</u> object from the cursor. A MarkTime object is identified by a date/time where it is displaying.
- <u>MarkZoneFromPoint</u> property gets the <u>MarkZone</u> object from the cursor. A MarkZone object is identified by a starting date/time and ending date/time where it is displaying.
- <u>NonworkingTimeFromPoint</u> property indicates the <u>NonworkingTime</u> object from the cursor.
- <u>PictureFromPoint</u> property indicates the key of the picture from the cursor.
- <u>TimeScaleFromPoint</u> property returns the <u>TimeScale</u> object from the cursor.

All ... FromPoint properties that returns an Object, may return Nothing, Empty or NULL, if no

object is found.

property Schedule.TimeScales as TimeScales

Gets the schedule's time scales collection.

Туре	Description
<u>TimeScales</u>	The TimeScales object which holds a collection of <u>TimeScale</u> objects.

By default, the control adds a time scale, that can be accessed using TimeScales(0) property. The control handles one or more time scales. Each time scale can display a different time zone, and can be aligned to any side of the schedule view. The TimeScales collection is accessible through the TimeScales property of the control. The <u>TimeScaleFromPoint</u> method gets the TimeScale object from the cursor. The <u>TimeScalesFont</u> defines the font to display the control's time scales. The <u>TimeZone</u> property defines the time zone of the time scale. The <u>AllowMoveTimeScale</u> property indicates the keys the user can move at runtime the time scale from a side to another. The <u>AllowResizeTimeScale</u> property indicates the keys the user can resize at runtime the time scale. Use the <u>Add</u> method of the TimeScales object to add a new time scale to the control. The <u>AlignLeft</u> property aligns the time scale to the left or to the right side of the schedule view.

The <u>DayStartTime</u> property defines the starting time of the day, and the <u>DayEndTime</u> property defines the ending time of the day.

The following screen show shows the control with two time-scales:



property Schedule.ToolTipDelay as Long

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

Туре	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 <u>ToolTipPopDelay</u> property specifies the period in ms of time the ToolTip remains visible if the mouse pointer is stationary within a control. Use the <u>ToolTipWidth</u> property to specify the width of the tooltip window. Use the <u>ToolTipFont</u> property to assign a font for the control's tooltip. Use the <u>Background(exToolTipAppearance)</u> property indicates the visual appearance of the borders of the tooltips. Use the <u>Background(exToolTipBackColor)</u> property indicates the tooltip's background color. Use the <u>Background(exToolTipForeColor)</u> property indicates the tooltip's foreground color.

property Schedule.ToolTipFont as IFontDisp

Retrieves or sets the tooltip's font.

Туре	Description
IFontDisp	A Font object used to paint the control's tooltip.

Use the ToolTipFont property to change the tooltip's font. The ForeColor property indicates the foreground color to show the captions or labels.

The control supports the following Font properties:

- Font property, that specifies the control's font, including the Calendar's font
- <u>EventsFont</u> property, indicates the font to show the captions and labels on Event/Appointment objects
- <u>TimeScaleFont</u> property, specifies the font to display the labels on the control's time scales
- ToolTipFont property specifies the font to display the tooltip being shown when the cursor is hovering an part of the control.

property Schedule.ToolTipPopDelay as Long

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

Туре	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 <u>ToolTipDelay</u> property specifies the time in ms that passes before the ToolTip appears. Use the <u>ToolTipWidth</u> property to specify the width of the tooltip window. Use the <u>ToolTipFont</u> property to assign a font for the control's tooltip. Use the <u>Background(exToolTipAppearance)</u> property indicates the visual appearance of the borders of the tooltips. Use the <u>Background(exToolTipBackColor)</u> property indicates the tooltip's background color. Use the <u>Background(exToolTipForeColor)</u> property indicates the tooltip's foreground color. Use the <u>ShowToolTip</u> method to display a custom tooltip.

property Schedule.ToolTipWidth as Long

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

Туре	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 <u>ToolTipPopDelay</u> property specifies the period in ms of time the ToolTip remains visible if the mouse pointer is stationary within a control. The <u>ToolTipDelay</u> property specifies the time in ms that passes before the ToolTip appears. Use the <u>Background(exToolTipAppearance)</u> property indicates the visual appearance of the borders of the tooltips. Use the <u>Background(exToolTipBackColor)</u> property indicates the tooltip's background color. Use the <u>Background(exToolTipForeColor)</u> property indicates the tooltip's foreground color. Use the <u>ShowToolTip</u> method to display a custom tooltip. Use the <u>ToolTipFont</u> property to assign a font for the control's tooltip.

method Schedule.Undo ()

Performs the last Undo operation.

Туре

Description

The Undo method undoes the last control operation. The <u>AllowUndoRedo</u> property enables or disables the Undo/Redo feature. The <u>CanUndo</u> method indicates whether the control can perform an Undo operation. The <u>Redo</u> redoes the next action in the control's redo queue. The <u>UndoRedoQueueLength</u> property gets or sets the maximum number of Undo/Redo actions that may be stored to the control's queue, or in other words how many operations the control's Undo/Redo manager may store.

The records of the Undo/Redo queue may contain actions in the following format:

- "AddEvent; EVENTID", indicates that a new calendar-event has been created
- "**RemoveEvent**; EVENTID", indicates that an calendar-event has been removed
- "MoveEvent; EVENTID", indicates that an calendar-event has been moved or resized
- "UpdateEvent; EVENTID", indicates that one or more properties of the calendar-event has been updated, using the <u>StartUpdateEvent</u> / <u>EndUpdateEvent</u> methods

Also, the Undo/Redo queue may include:

- "StartBlock", specifies that a block of operations begins (initiated by <u>StartBlockUndoRedo</u> method)
- "EndBlock", specifies that a block of operations ends (initiated by EndBlockUndoRedo method)

The LayoutStartChanging(exUndo/exRedo) / LayoutEndChanging(exUndo/exRedo) event notifies your application whenever an Undo/Redo operation is performed. The UndoListAction property lists the Undo actions that can be performed in the control. The RedoListAction property lists the Redo actions that can be performed in the control. Use the UndoRemoveAction method to remove the last actions from the undo queue. The RedoRemoveAction method removes the first action to be performed if the Redo method is invoked.

property Schedule.UndoListAction ([Action as Variant], [Count as Variant]) as String

Lists the Undo actions that can be performed on the control.

Туре	Description
	[optional] A long expression that specifies the action being listed. If missing or -1, all actions are listed.
	The Action parameter can be one of the following:
Action as Variant	 exUndoRedoAddEvent(13) ~ "AddEvent;EVENTID", indicates that a new calendar-event has been created exUndoRedoRemoveEvent(14) ~ "RemoveEvent;EVENTID", indicates that an calendar-event has been removed exUndoRedoMoveEvent(15) ~ "MoveEvent;EVENTID", indicates that an calendar-event has been moved or resized exUndoRedoUpdateEvent(16) ~ "UpdateEvent;EVENTID", indicates that one or more properties of the calendar-event has been updated, using the <u>StartUpdateEvent</u> / EndUpdateEvent methods
	For instance, UndoListAction(12) shows only AddEvent actions in the undo stack.
Count as Variant	[optional] A long expression that indicates the number of actions being listed. If missing or -1, all actions are listed. For instance, UndoListAction(12,1) shows only the last AddEvent action being added to the undo stack
String	A String expression that lists the Undo actions that may be performed.

The UndoListAction property lists the Undo actions that can be performed in the control. The <u>AllowUndoRedo</u> property enables or disables the Undo/Redo feature. The <u>RedoListAction</u> property lists the Redo actions that can be performed in the control. Use the <u>UndoRemoveAction</u> method to remove the last actions from the undo queue. The <u>RedoRemoveAction</u> method removes the first action to be performed if the Redo method is invoked. The <u>LayoutStartChanging</u>(exUndo/exRedo) / <u>LayoutEndChanging</u>(exUndo/exRedo) event notifies your application whenever an Undo/Redo operation is performed. The records of the Undo/Redo queue may contain actions in the following format:

- "AddEvent; EVENTID", indicates that a new calendar-event has been created
- "RemoveEvent; EVENTID", indicates that an calendar-event has been removed
- "MoveEvent; EVENTID", indicates that an calendar-event has been moved or resized
- "UpdateEvent;EVENTID", indicates that one or more properties of the calendar-event has been updated, using the <u>StartUpdateEvent</u> / <u>EndUpdateEvent</u> methods

Also, the Undo/Redo queue may include:

- "StartBlock", specifies that a block of operations begins (initiated by <u>StartBlockUndoRedo</u> method)
- "EndBlock", specifies that a block of operations ends (initiated by EndBlockUndoRedo method)

Here's a sample how the result of UndoListAction method looks like:

StartBlock	
MoveEvent;3	
AddEvent;3	
EndBlock	
MoveEvent;B	
UpdateLink;Akak	
UpdateEvent;2	
AddEvent;2	
UpdateEvent;1	
AddEvent;1	

property Schedule.UndoRedoQueueLength as Long

Gets or sets the maximum number of Undo/Redo actions that may be stored to the control's queue.

Туре	Description
Long	A Long expression that specifies the length of the Undo/Redo queue. If -1, the queue is unlimited, 0 allows no entries in the Undo/Redo queue (Undo/Redo is disabled).

By default, the UndoRedoQueueLength property is -1. The <u>AllowUndoRedo</u> property enables or disables the Undo/Redo feature. Use the UndoRedoQueueLength property to specify the number of entries that Undo/Redo queue may store. For instance, if the UndoRedoQueueLength property is 1, the control retains only the last chart operation. Changing the UndoRedoQueueLength property may change the current Undo/Redo queue based on the new length. The length being specified, does not affect the blocks in the queue. A block may hold multiple Undo/Redo actions. Use the <u>GroupUndoRedoActions</u> method to group two or more entries in the Undo/Redo queue in a single block, so when a next Undo/Redo operation is performed, multiple actions may occur. For instance, moving several calendar-events in the same time (multiple calendar-events selection) is already recorded as a single block.

method Schedule.UndoRemoveAction ([Action as Variant], [Count as Variant])

Removes the last undo actions that can be performed on the control.

Туре	Description
	[optional] A long expression that specifies the action being remove. If missing or -1, all actions are removed.
	The Action parameter can be one of the following:
Action as Variant	 exUndoRedoAddEvent(13) ~ "AddEvent;EVENTID", indicates that a new calendar-event has been created exUndoRedoRemoveEvent(14) ~ "RemoveEvent;EVENTID", indicates that an calendar-event has been removed exUndoRedoMoveEvent(15) ~ "MoveEvent;EVENTID", indicates that an calendar-event has been moved or resized exUndoRedoUpdateEvent(16) ~ "UpdateEvent;EVENTID", indicates that one or more properties of the calendar-event has been updated, using the <u>StartUpdateEvent</u> / <u>EndUpdateEvent</u> methods
	For instance, UndoRemoveAction(12) removes only AddEvent actions from the undo stack.
Count as Variant	[optional] A long expression that indicates the number of actions to remove. If missing or -1, all actions are removed. For instance, UndoRemoveAction(12,1) removes only the last AddEvent action from the undo stack
	•

Use the UndoRemoveAction method to remove the last action from the undo queue. Use the UndoRemoveAction() (with no parameters) to remove all undo actions. The <u>RedoRemoveAction</u> method removes the first action to be performed if the Redo method is invoked. The <u>AllowUndoRedo</u> property enables or disables the Undo/Redo feature. The <u>UndoListAction</u> property lists the Undo actions that can be performed in the control. The <u>RedoListAction</u> property lists the Redo actions that can be performed in the control. The <u>LayoutStartChanging</u>(exUndo/exRedo) / <u>LayoutEndChanging</u>(exUndo/exRedo) event notifies your application whenever an Undo/Redo operation is performed.

The records of the Undo/Redo queue may contain actions in the following format:

- "AddEvent; EVENTID", indicates that a new calendar-event has been created
- "RemoveEvent; EVENTID", indicates that an calendar-event has been removed
- "**MoveEvent**; EVENTID", indicates that an calendar-event has been moved or resized
- "UpdateEvent; EVENTID", indicates that one or more properties of the calendar-event has been updated, using the <u>StartUpdateEvent</u> / <u>EndUpdateEvent</u> methods

Also, the Undo/Redo queue may include:

- "StartBlock", specifies that a block of operations begins (initiated by <u>StartBlockUndoRedo</u> method)
- "EndBlock", specifies that a block of operations ends (initiated by EndBlockUndoRedo method)

property Schedule.UpdateEventsLabel as String

Specifies the label to be shown while moving or resizing the events.

Туре	Description
String	A String expression that defines the label to be displayed when the user moves or resizes the events at runtime. The UpdateEventsLabel supports extended HTML format as explained bellow.

By default, the UpdateEventsLabel property is: "<%=%256%>
<%=((1:=int(0:= (date(%2)-date(%1)))) != 0 ? (=:1 + ' day(s)') : ") + (=:1 ? ' ' : ") + ((1:=int(0:=((=:0 - =:1 +) 1/24/60/60/2*24))) != 0 ? =:1 + ' hour(s)' : ") + (=:1 ? ' ' : ") + ((1:=round((=:0 - =:1)*60))) != 0 ? =:1 + ' min(s)' : ")%>" which displays the margins of the event being moved or resized on the first line, while on the second line it displays the duration of the event. The UpdateEventsLabel property indicates the HTML format to be shown on the label when the user moves or resizes the events. The UpdateEventsLabelAlign property aligns the label being shown when the user moves or resizes event. The AllowMoveEvent property indicates the combination of the keys to let user moves the events. The AllowResizeEvent property indicates the combination of the keys to let user resizes the events. The AllowCreateEvent property indicates the combination of keys that allows the user to create new events in the control. The <u>Background(exScheduleUpdateEventsBackColor)</u> and Background(exScheduleUpdateEventsForeColor) specifies the visual appearance of the event being created. The UpdateEvent event occurs once an event is resized or moved. The DefaultEventPadding property indicates the padding of the labels on the event, relative to event's borders.

Here's a few samples:

- "new", simple new text is shown.
- "<a no>title", displays a clickable text such as <u>title</u>, and <u>AnchorClick</u> can be used to determine whether the *no* anchor has been clicked.
- "<a>pic1:32", displays a click able image, the <u>AnchorClick</u> can be used to determine whether the anchor has been clicked. We would recommend using the <u>Pictures</u> or <u>ExtraPictures</u> property to assign pictures to an event.
- "<%=%256%>", displays the event's start and end points in a short format.
- "<%=%257%>", displays the event's margins in a long format.
- "Start: <%=%1%>
End: <%=%2%>", displays the starting margin of the even on the first line, while on the second line it displays the ending point of the event.
- "<%=%256%>
Caption: <%=%5%>", displays the event's margins in short format on the first line, and on the second line it displays the event's <u>Caption</u> property. The caption shown on the event's body is automatically updated once the event is moved to a new position or the event's Caption is changed.

- "<%=%256%>
<%=%264? `repetitive event`:``%>" displays automatically the "repetitive event" for repetitive events, or when the event's <u>Repetitive</u> property is not empty and valid
- "Duration: <%=((1:=int(0:= (date(%2)-date(%1)))) != 0 ? (=:1 + ' day(s)') : ") + (=:1 ? ' ' : ") + ((1:=int(0:=((=:0 =:1 + 1/24/60/60/2)*24))) != 0 ? =:1 + ' hour(s)' : ") + (=:1 ? ' ' : ") + ((1:=round((=:0 =:1)*60)) != 0 ? =:1 + ' min(s)' : ")%>" displays the duration of the event in days, hours and minutes.
- "<%=%256%>
Duration: <%=((1:=int(0:= (date(%2)-date(%1)))) != 0 ? (=:1 + ' day(s)') : ") + (=:1 ? ' ' : ") + ((1:=int(0:=((=:0 =:1 + 1/24/60/60/2)*24))) != 0 ? =:1 + ' hour(s)' : ") + (=:1 ? ' ' : ") + ((1:=round((=:0 =:1)*60)) != 0 ? =:1 + ' min(s)' : ")%>" displays the event's margins on the first line and the duration of the event in days, hours and minutes, on the second line
- "<%=%><%=%5%>
<%=%256%>", displays the event's Caption on first line(s), following by the event's Start/End margins in short date-time format. The <%=%> prefix forces the expression to be re-evaluated and apply any HTML tag found. For instance, %5 indicates the event's Caption property, and if it contains HTML tags they will be applied as is, instead displaying them as a plain text. Any expression that starts with "<%=%>" is re- evaluated and its result is displayed in HTML format (available starting with the version 12.2)

The <u>EventKnowPropertyEnum</u> defines the %identifiers that can be used in formula <%=FORMULA%>. For instance, the CreateEventLabel property on "Start:<%=time(%1) replace `AM` with ``%>" displays the time when the event starts with no AM time indicators.

The property supports the following identifiers. These identifiers can be used in FORMULA format:

- %1, Indicates the starting date/time of the event as DATE type, equivalent with <u>Start</u> property
- %2, Indicates the ending date/time of the event as DATE type, equivalent with <u>End</u> property
- %3, Indicates if the current event is an all day event as BOOL type, equivalent with <u>AllDayEvent</u> property
- %4, Indicates the identifier of the event's group, as LONG type, equivalent with <u>GroupID</u> property.
- **%5**, Indicates the caption of the event, as STRING expression, equivalent with <u>Caption</u> property.
- %6, Indicates the extra data associated with the event, as VARIANT type, equivalent with <u>UserData</u> property.
- %7, Gets or sets the duration of the event as FLOAT expression. Above you can find how you can display the duration of the event in hours, minutes...
- %8, Specifies the repetitive expression of the event, equivalent with Repetitive

property.

- %256, Gets the margins of the event in a short format, as a STRING expression. The <u>ShortDateFormat</u> property defines the short date format. The <u>ShortTimeFormat</u> property defines the short time format.
- %257, Gets the margins of the event in a long format, as a STRING expression. The <u>LongDateFormat</u> property defines the long date format. The <u>LongTimeFormat</u> property defines the long time format.
- %258, Gets the starting date (not including the time) of the current event, as a DATE type.
- %259, Gets the starting time (not including the date) of the current event, as DATE type from 0 to 1.
- %260, Gets the ending date (not including the time) of the current event, as a DATE type.
- %261, Gets the ending time (not including the date) of the current event, as DATE type from 0 to 1.
- %262, Gets the label of the owner group, as STRING expression, equivalent with <u>Caption</u> property of the Group's event.
- %263, Gets the title of the owner group, as STRING expression, equivalent with <u>Title</u> property of the Group's event.
- %264, Indicates if the current event is a repetitive event, as BOOL type. This flag returns TRUE, if the <u>Repetitive</u> property is not empty, and the expression is valid.

The FORMULA, is identified by <%=FORMULA%>, and supports 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#, #4/1/2002#:1 ; #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:

- \circ 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
- $\circ~$ 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)
 - I 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
- Itrim (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 property supports the following built-in HTML tags:

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

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

- exp, stores the plain text to be shown once the user clicks the anchor, such as "<a ;exp=show lines>"
- e64, encodes in BASE64 the HTML text to be shown once the user clicks the anchor, such as "<a/li>

;e64=gA8ABmABnABjABvABshIAOQAEAAHAAGESikWio+ABzABohp3iELABpABu " that displays show lines- in gray when the user clicks the + anchor. The "gA8ABmABnABjABvABshIAOQAEAAHAAGESikWio+ABzABohp3iELABpABuABljY string encodes the "<fgcolor 808080>show lines<a>-</fgcolor>" The Decode64Text/Encode64Text methods of the eXPrint can be used to decode/encode e64 fields. Any ex-HTML caption can be transformed to an expandable-caption, by inserting the anchor ex-HTML tag. For instance, "<solidline>Header</solidline>
Line1<r><a ;exp=show lines>+
Line2
Line3" shows the Header in underlined and bold on the first line and Line1, Line2, Line3 on the rest. The "show lines" is shown instead of Line1, Line2, Line3 once the user clicks the + sign.

- ... displays portions of text with a different font and/or different size. For instance, the "bit" draws the bit text using the Tahoma font, on size 12 pt. If the name of the font is missing, and instead size is present, the current font is used with a different size. For instance, "bit" displays the bit text using the current font, but with a different size.
- **<fgcolor rrggbb>** ... **</fgcolor>** or **<**fgcolor=rrggbb> ... **</**fgcolor> displays text with a specified foreground color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- <bgcolor rrggbb> ... </bgcolor> or <bgcolor=rrggbb> ... </bgcolor> displays text with a specified background color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- <solidline rrggbb> ... </solidline> or <solidline=rrggbb> ... </solidline> draws a solidline 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
- number[:width] 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.
- key[:width] 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 FFFFF;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 Schedule.UpdateEventsLabelAlign as ContentAlignmentEnum

Specifies the alignment of the label to be shown while moving or resizing events.

Туре	Description
<u>ContentAlignmentEnum</u>	A ContentAlignmentEnum expression that specifies the alignment of the label being shown when the user moves or resizes the events.

The UpdateEventsLabelAlign property aligns the label being shown when the user moves or resizes the events. The <u>CreateEventLabelAlign</u> property aligns the label being shown when the user creates a new event.

property Schedule.UseVisualTheme as UIVisualThemeEnum

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

Туре	Description
<u>UIVisualThemeEnum</u>	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 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 VisualDesign property may be used to change the visual aspect of the entire control at design mode.

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 Schedule.Version as String

Retrieves the control's version.

Туре	Description
String	A String expression that specifies the control's version.

The version property specifies the control's version.

property Schedule.VerticalScrollWheel as Double

Indicates the distance to scroll using the mouse wheel.

Туре	Description
Double	A double expression that indicates the distance to scroll using the mouse wheel. A positive value is multiplied with the font's height, while a negative value indicates a fixed distance.

By default, the VerticalScrollWheel property is 1.0. The VerticalScrollWheel property determines the distance to scroll using the mouse wheel by multiplying the value with the height of the current font, if the value is positive, else if the value is negative it indicates a fixed distance. The Font property indicates the control's font. Use the VerticalScrollWheel property to programmatically specify the increment to scroll the control's content by rotating the mouse wheel. The VerticalScrollWheel property on 0, indicates that no scroll is performed when user rotates the mouse wheel.

property Schedule.VisualAppearance as Appearance

Retrieves the control's appearance.

Туре	Description
Appearance	An Appearance object that holds the EBN objects, that can be applied on any UI part of the control.

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 control provides the <u>VisualDesign</u> property that allows you to easily change the control's visual appearance at design mode. Also, the VisualDesign property can be used at runtime to specify a visual appearance, by setting the VisualDesign property with a new generated value. The <u>UseVisualTheme</u> property indicates whether the current visual theme is applied to parts of the control.

Use the <u>Add</u> 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.

Here's a screen shot skins a few UI parts of the component, using the EBN objects :



property Schedule.VisualDesign as String

Invokes the control's VisualAppearance designer.

Туре	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 <u>EBN</u> 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 <u>VisualAppearance</u> 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.
- The /WPF version does not provide a VisualAppearance designer, instead you can use the values being generated by the /COM or /NET to apply the same visual appearance.
- Click here
 ■ to watch a movie on how you define the control's visual appearance using the XP-Theme
- Click here I to watch a movie on how you define the control's visual appearance using the 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):
Property Pages									X
Color Font Template The Visual Design page all click the EBN and drag to	Visual De lows you to the part of t	esign change the he control y	e visual ap rou want t	opeara to cha	ince for differer nge its visual a	it parts of the control. I ppearance.	Right click or drag any EBN	l files to the left panel, ar	nd next
							Wednesday, June 27, 20	12	-
							Groups		
	•	June 201	12	•	08:00 AM				
	Su I			Sa	:30	8:30 AM - 10:45 AM			
	22		1	2	:45 09:00 AM	normal	8:45 AM - 11:00 AM	0-00 AM 11-15 AM	
	23 3	4 5 6	7 8	9	:15		selected	5.00 AM - 11.15 AM	
	24 10	11 12 13	14 15	16	:30			normal	1
	25 17	18 19 20	21 22	23	10:00 AM				
	26 24	25 26 27	28 29	30	:15				
	27	0			:30				
					11:00 AM		-		
					:15				
				_	:45			control	s tooltip
		Today			•		III		
							OK		á mel i
							OK		Арру

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

olor he Vis	Font ual Des	Template	Vis ows y	ual [ou to	Desig	n ange	the	visua	al app	pear	nce for differe	t parts of the control.	Right click or drag any E	BN files to the left panel, ar	id next
01					une (John	ior ye			Jone			Wednesday, June 27, Groups	2012	102
02		×			J	une	2013	2			08:00 AM				
03 04 (05 (06		×	22 23 24 25 26 27	3 10 17 24	4 11 18 25	Tu 5 12 19 26	6 13 20 27	Th 7 14 21 28	Fr 1 8 15 22 29	2 9 16 23 30	:30 :45 09:00 AM :15 :30 :45 10:00 AM :15 :30 :45 11:00 AM	8:30 AM - 10:45 AM normal	8:45 AM - 11:00 AM selected	9:00 AM - 11:15 AM normal	
					(Тос	day]			:15 :30 :45		 ОК	Cancel	toottip

TimeScale object

The TimeScale object displays the time-scale in the control. The control handles one or more time scales. Each time scale can display a different time zone, and can be aligned to any side of the schedule view. The <u>TimeZone</u> property defines the time zone of the time scale. The <u>MajorTimeRuler</u> property indicates the time to increment the major rulers, while the <u>MinorTimeRuler</u> property specifies the time to increment the minor rulers. The <u>DayStartTime</u> property defines the starting time of the day, and the <u>DayEndTime</u> property defines the day.

The TimeScale object shows the control's time scale in red:



The TimeScale object supports the following properties and methods:

Name	Description
<u>AlignLeft</u>	Specifies whether the time scale is aligned to left or to the right of the scheduler.
AllowResize	Specifies whether the user can resize the TimeScale object.
BackColor	Specifies the TimeScale's background color.
Caption	Indicates the caption to be displayed on the TimeScale's header.

CaptionAlign	Indicates the alignment of the TimeScale's caption.
CaptionBackColor	Specifies the background color for the TimeScale's caption.
CaptionForeColor	Specifies the foreground color for the TimeScale's caption.
<u>ForeColor</u>	Specifies the TimeScale's foreground color.
Index	Indicates the index of the time scale object in the TimeScales collection.
<u>MajorLabelColor</u>	Specifies the foreground color to display the labels of major rulers.
<u>MajorTimeLabel</u>	Indicates the label to be displayed on the major ruler of the current TimeScale object.
<u>MajorTimeLabelPlainText</u>	Specifies whether the major label is a plain text or a formatted HTML text.
<u>MajorTimeRuler</u>	Indicates the major increment for the current time scale.
MaxWidth	Gets or sets a value that indicates the maximum width for the current TimeScale object.
<u>MinorLabelColor</u>	Specifies the foreground color to display the labels of minor rulers.
<u>MinorTimeLabel</u>	Indicates the label to be displayed on the minor ruler of the current TimeScale object.
<u>MinorTimeLabelPlainText</u>	Specifies whether the minor label is a plain text or a formatted HTML text.
<u>MinorTimeRuler</u>	Indicates the minor increment for the current time scale.
<u>MinWidth</u>	Gets or sets a value that indicates the minimum width for the current TimeScale object.
Position	Gets or sets the position of the current time scale.
RulerBackColor	Specifies the background color for TimeScale's ruler.
<u>TimeZone</u>	Indicates the time zone for the current time scale.
<u>ToolTip</u>	Indicates the tooltip of the TimeScale object.
<u>UserData</u>	Indicates any extra data associated with the TimeScale object.
Visible	Specifies whether the TimeScale is visible or hidden.
Width	Gets or sets a value that indicates the TimeScale's width, in pixels.

property TimeScale.AlignLeft as Boolean

Specifies whether the time scale is aligned to left or to the right of the scheduler.

Туре	Description
Boolean	A Boolean expression that specifies whether the time scale is aligned on the left or right side of the schedule view.

By default, the AlignLeft property is True, which indicates that the time scale is aligned to the left side of the schedule view. The AlignLeft property can be used to programmatically change the time scale alignment. The <u>AllowMoveTimeScale</u> property indicates the keys the user can move at runtime the time scale from a side to another. The <u>Position</u> property indicates the position of the time scale as they are displayed. The AlignLeft and <u>Position</u> properties may be changed if the user moves the time scale position to a new side, while the <u>AllowMoveTimeScale</u> property is not exDisallow (0).

property TimeScale.AllowResize as Boolean

Specifies whether the user can resize the TimeScale object.

Туре	Description
Boolean	A Boolean expression that specifies whether the user can resize the current time scale at runtime.

By default, the AllowResize property is True. The <u>AllowResizeTimeScale</u> property indicates the keys the user can resize at runtime the time scale. The AllowResize property may specify whether a time scale is resizable or not, while AllowResizeTimeScale property on exDisallow may specify that no time scale can be resized at runtime. The <u>Width</u> property of the time scale indicates the width in pixels of the time scale. The <u>MinWidth</u> property indicates the minimum width for the time scale, and the <u>MaxWidth</u> indicates the maximum size of the time scale.

The LayoutStartChanging(exScheduleResizeTimeScale) and

LayoutEndChanging(exScheduleResizeTimeScale) properties notifies your application once the user resizes a time scale. The <u>Width</u> property may be changed while the user resizes the time scale.

property TimeScale.BackColor as Color

Specifies the TimeScale's background color.

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

The BackColor property is initialized with the value of the <u>Background</u>(exScheduleTimeScaleBackColor) property. By default, the <u>Background</u>(exScheduleTimeScaleBackColor) property defines the background color to show the control's time scales.

The time scale supports the following background properties:

- The BackColor property changes the time scale's background.
- The <u>CaptionBackColor</u> property defines the color to show caption of the timescale. The <u>Caption</u> property defines the HTML caption to be displayed on the top of the time scale.
- The <u>RulerBackColor</u> property specifies the background color for the rulers part of the time scale. The <u>MajorTimeLabel</u> property defines the time label to be shown on the major rulers, while the <u>MinorTimeLabel</u> property specifies the time label to be shown on minor rulers.

All background colors can display a solid color as well as an EBN object.

The following sample shows the time scale using different background colors:



property TimeScale.Caption as String

Indicates the caption to be displayed on the TimeScale's header.

Туре	Description
String	A String expression that specifies the HTML caption to be displayed on the top side of the time scale.

The Caption property specifies the HTML caption to be displayed on the top side of the time scale. The <u>CaptionAlign</u> property aligns the caption. The <u>MajorTimeLabel</u> property defines the time label to be shown on the major rulers, while the <u>MinorTimeLabel</u> property specifies the time label to be shown on minor rulers. The <u>TimeZone</u> property can be used to programmatically update the time zone. The <u>CaptionBackColor</u> property defines the color to show caption's background of the timescale. The <u>CaptionForeColor</u> property defines the color to show caption of the timescale.

The Caption property supports the following HTML tags:

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

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

 exp, stores the plain text to be shown once the user clicks the anchor, such as "<a ;exp=show lines>"

;e64=gA8ABmABnABjABvABshIAOQAEAAHAAGESikWio+ABzABohp3iELABpABu " that displays show lines- in gray when the user clicks the + anchor. The "gA8ABmABnABjABvABshIAOQAEAAHAAGESikWio+ABzABohp3iELABpABuABljY string encodes the "<fgcolor 808080>show lines<a>-</fgcolor>" The Decode64Text/Encode64Text methods of the eXPrint can be used to decode/encode e64 fields. Any ex-HTML caption can be transformed to an expandable-caption, by inserting the anchor ex-HTML tag. For instance, "<solidline>Header</solidline>
Line1<r><a ;exp=show lines>+
Line2
Line3" shows the Header in underlined and bold on the first line and Line1, Line2, Line3 on the rest. The "show lines" is shown instead of Line1, Line2, Line3 once the user clicks the + sign.

- ... displays portions of text with a different font and/or different size. For instance, the "bit" draws the bit text using the Tahoma font, on size 12 pt. If the name of the font is missing, and instead size is present, the current font is used with a different size. For instance, "bit" displays the bit text using the current font, but with a different size.
- <fgcolor rrggbb> ... </fgcolor> or <fgcolor=rrggbb> ... </fgcolor> displays text with a specified foreground color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- <bgcolor rrggbb> ... </bgcolor> or <bgcolor=rrggbb> ... </bgcolor> displays text with a specified background color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<solidline rrggbb>** ... **</solidline>** or **<**solidline=rrggbb> ... **<**/solidline> draws a solidline 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
- number[:width] 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.
- key[:width] inserts a custom size picture into the text being previously loaded using the HTMLPicture property. The Key parameter indicates the key of the picture being displayed. The Width parameter indicates a custom size, if you require to

stretch the picture, else the original size of the picture is used.

- & glyph characters as & amp; (&), &It; (<), > (>), &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 &It;b>bold&It;/b>
- **<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 FFFFF;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 TimeScale.CaptionAlign as ContentAlignmentEnum

Indicates the alignment of the TimeScale's caption.

Туре

```
Description
```

<u>ContentAlignmentEnum</u>

A ContentAlignmentEnum expression that specifies the alignment of the caption in the time scale header.

By default, the CaptionAlign property is exMiddleLeft. The <u>Caption</u> property specifies the HTML caption to be displayed on the top side of the time scale. The <u>CaptionBackColor</u> property defines the color to show caption's background of the timescale. The <u>CaptionForeColor</u> property defines the color to show caption of the timescale.

property TimeScale.CaptionBackColor as Color

Specifies the background color for the TimeScale's caption.

Туре	Description
Color	A Color expression that specifies the background color to show the caption part of the time scale. The last 7 bits in the high significant byte of the color to indicates the identifier of the skin being used. Use the <u>Add</u> 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.

By default, the CaptionBackColor property 0. The CaptionBackColor property defines the visual appearance of the caption part of the time scale, if it is not zero. If the CaptionBackColor property is zero (default), the <u>BackColor</u> property indicates the time scale's background color.

The time scale supports the following background properties:

- The <u>BackColor</u> property changes the time scale's background.
- The CaptionBackColor property defines the color to show caption of the timescale. The <u>Caption</u> property defines the HTML caption to be displayed on the top of the time scale.
- The <u>RulerBackColor</u> property specifies the background color for the rulers part of the time scale. The <u>MajorTimeLabel</u> property defines the time label to be shown on the major rulers, while the <u>MinorTimeLabel</u> property specifies the time label to be shown on minor rulers.

All background colors can display a solid color as well as an EBN object.

The following sample shows the time scale using different background colors:



property TimeScale.CaptionForeColor as Color

Specifies the foreground color for the TimeScale's caption.

Туре	Description
Color	A Color expression that specifies the foreground color to show the caption part of the time scale.

By default, the CaptionForeColor property 0. The CaptionForeColor property defines the color to show the caption of the time scale, if it is not zero. If the CaptionForeColor property is zero (default), the <u>ForeColor</u> property indicates the time scale's foreground color.

The time scale supports the following foreground properties:

- The <u>ForeColor</u> property changes the time scale's foreground.
- The CaptionForeColor property defines the color to show caption of the timescale. The <u>Caption</u> property defines the HTML caption to be displayed on the top of the time scale.
- The <u>MajorLabelColor</u> property indicates the color to show the labels on major rulers. The <u>MajorTimeLabel</u> property defines the time label to be shown on the major rulers.
- The <u>MinorLabelColor</u> property indicates the color to show the labels on minor rulers. The <u>MinorTimeLabel</u> property defines the time label to be shown on the minor rulers.

property TimeScale.ForeColor as Color

Specifies the TimeScale's foreground color.

Туре	Description
Color	A Color expression that specifies the foreground color to show the time scale.

The time scale supports the following foreground properties:

- The ForeColor property changes the time scale's foreground.
- The <u>CaptionForeColor</u> property defines the color to show caption of the timescale. The <u>Caption</u> property defines the HTML caption to be displayed on the top of the time scale.
- The <u>MajorLabelColor</u> property indicates the color to show the labels on major rulers. The <u>MajorTimeLabel</u> property defines the time label to be shown on the major rulers.
- The <u>MinorLabelColor</u> property indicates the color to show the labels on minor rulers. The <u>MinorTimeLabel</u> property defines the time label to be shown on the minor rulers.

property TimeScale.Index as Long

Indicates the index of the time scale object in the TimeScales collection.

Туре	Description
Long	A long expression that defines the time scale.

The Index property specifies the index of the time scale in the <u>TimeScales</u> collection. By default, the control adds a default time scale that can be accesses using the property TimeScales(0). The <u>Item</u> property gets the time scale using its index or time zone. The <u>TimeZone</u> property can be used to programmatically update the time zone. The <u>Count</u> property indicates the number of time scales in the control. The Index property goes from 0 to Count - 1.

property TimeScale.MajorLabelColor as Color

Specifies the foreground color to display the labels of major rulers.

Туре	Description
Color	A Color expression that specifies the color to show the labels on the major rulers.

The MajorLabelColor property indicates the color to show the labels on major rulers. The <u>MajorTimeLabel</u> property defines the time label to be shown on the major rulers, while the <u>MinorTimeLabel</u> property specifies the time label to be shown on minor rulers. The <u>MajorTimeLabelPlainText</u> property should be set on False, if the MajorTimeLabel property uses UI font attributes like , <fgcolor>, and so on. The <u>MajorTimeRuler</u> property indicates the time to show the major rulers.

property TimeScale.MajorTimeLabel as String

Indicates the label to be displayed on the major ruler of the current TimeScale object.

Туре	Description
String	A string expression that defines the extended HTML format to display the labels on the major rulers.

By default, the MajorTimeLabel property is "<%hh%>:<%nn%> <%AM/PM%>", which indicates that the major rulers displays the time using the AM/PM time indicators. The MajorTimeLabel property defines the time label to be shown on the major rulers, while the MinorTimeLabel property specifies the time label to be shown on minor rulers. The MajorLabelColor property indicates the color to show the labels on major rulers. The MajorTimeLabelPlainText property should be set on False, if the MajorTimeLabel property uses UI font attributes like , <fgcolor>, and so on. The MajorTimeRuler property indicates the major rulers.

The <u>TimeZone</u> property can be used to programmatically update the time zone. You can use the <u>ShortTimeFormat</u> property to define a 24-hours format, by removing the <%AM.PM%> TAG. The <u>Caption</u> property specifies the HTML caption to be displayed on the top side of the time scale.

The major lines can be changed using the following **<u>Background</u>** properties:

- The Background(exScheduleTimeScaleMajorRulerColor) specifies the color to show the lines for major rulers (This option changes the lines being shown in the time scale header)
- The Background(exScheduleTimeScaleMajorRulerStyle) property indicates a <u>LinesStyleEnum</u> expression that determines the style of lines to be shown on major rulers (This option changes the lines being shown in the time scale header)
- The Background(exScheduleMajorTimeScaleStyle) indicates a <u>LinesStyleEnum</u> expression that determines style of lines to be shown on the schedule view.
- The Background(exScheduleMajorTimeRulerColor) indicates the color to show the major lines on the schedule view

Here's a few samples on how you can use the MajorTimeLabel property:

- "", no label is shown on the major rulers
- "<%hh%>:<%nn%> <%AM/PM%>", hour and minute of the major, using the AM/PM time indicators.
- "<%hh%> <%AM/PM%>", hour using the AM/PM time indicators with a larger font. The <u>MajorTimeLabelPlainText</u> property must be set on False.

The MajorTimeLabel property supports the following HTML tags:

- <%hy%> Date displayed as the half of the year (1 to 2).
- <%loc_gg%> Indicates period/era using the current user regional and language settings.
- <%loc_sdate%> Indicates the date in the short format using the current user regional and language settings.
- <%loc_ldate%> Indicates the date in the long format using the current user regional and language settings.
- <%loc_dsep%> Indicates the date separator using the current user regional and language settings (/).
- <%h%> Hour in one or two digits, as needed (0 to 23).
- <%**hh**%> Hour in two digits (00 to 23).
- <%**n**%> Minute in one or two digits, as needed (0 to 59).
- <%**nn**%> Minute in two digits (00 to 59).
- <%s%> Second in one or two digits, as needed (0 to 59).
- <%**ss**%> Second in two digits (00 to 59).
- <%AM/PM%> Twelve-hour clock with the uppercase letters "AM" or "PM", as appropriate. (Use the <u>AMPM</u> property to specify the name of the AM and PM indicators). You can use the <%loc_AM/PM%> that indicates the time marker such as AM or PM using the current user regional and language settings. You can use <%loc_A/P%> that indicates the one character time marker such as A or P using the current user regional and language settings.
- <%loc_AM/PM%> Indicates the time marker such as AM or PM using the current user regional and language settings.
- <%loc_A/P%> Indicates the one character time marker such as A or P using the current user regional and language settings.
- <%loc_time%> Indicates the time using the current user regional and language settings.
- <%loc_time24%> Indicates the time in 24 hours format without a time marker using the current user regional and language settings.
- <%loc_tsep%> indicates the time separator using the current user regional and language settings (:).

The MajorTimeLabel property supports the following HTML tags (The <u>MajorTimeLabelPlainText</u> property must be set on False, else the following UI attributes are not shown) :

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

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

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

- exp, stores the plain text to be shown once the user clicks the anchor, such as "<a ;exp=show lines>"
- e64, encodes in BASE64 the HTML text to be shown once the user clicks the anchor, such as "<a/li>

;e64=gA8ABmABnABjABvABshIAOQAEAAHAAGESikWio+ABzABohp3iELABpABu " that displays show lines- in gray when the user clicks the + anchor. The "gA8ABmABnABjABvABshIAOQAEAAHAAGESikWio+ABzABohp3iELABpABuABljY string encodes the "<fgcolor 808080>show lines<a>-</fgcolor>" The Decode64Text/Encode64Text methods of the eXPrint can be used to decode/encode e64 fields.

Any ex-HTML caption can be transformed to an expandable-caption, by inserting the anchor ex-HTML tag. For instance, "<solidline>Header</solidline>

Line1<r><a ;exp=show lines>+
Line2
Line3" shows the Header in underlined and bold on the first line and Line1, Line2, Line3 on the rest. The "show lines" is shown instead of Line1, Line2, Line3 once the user clicks the + sign.

- ... displays portions of text with a different font and/or different size. For instance, the "bit" draws the bit text using the Tahoma font, on size 12 pt. If the name of the font is missing, and instead size is present, the current font is used with a different size. For instance, "bit" displays the bit text using the current font, but with a different size.
- **<fgcolor rrggbb>** ... **</fgcolor>** or **<**fgcolor=rrggbb> ... **</**fgcolor> displays text with a specified foreground color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- <bgcolor rrggbb> ... </bgcolor> or <bgcolor=rrggbb> ... </bgcolor> displays text with a specified background color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- <solidline rrggbb> ... </solidline> or <solidline=rrggbb> ... </solidline> draws a solidline 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
- number[:width] 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.
- key[:width] 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 FFFFF;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:



<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 TimeScale.MajorTimeLabelPlainText as Boolean

Specifies whether the major label is a plain text or a formatted HTML text.

Туре	Description
Boolean	A Boolean expression that specifies whether the label on major rulers UI font attributes like , <fgcolor> are hidden or shown.</fgcolor>

By default, the MajorTimeLabelPlainText property is True, which indicates that no UI font attributes like , <fgcolor> are displayed when any of them is included in the <u>MajorTimeLabel</u> property. The <u>MajorTimeLabel</u> property defines the time label to be shown on the major rulers, while the <u>MinorTimeLabel</u> property specifies the time label to be shown on minor rulers. Using this property on False, may affect the control's performances. Do not set on False, unless you are using any of the specified UI font tags.

property TimeScale.MajorTimeRuler as String

Indicates the major increment for the current time scale.

l

Туре	Description
String	A String expression that specifies the time to show the next major ruler.

By default, the MajorTimeRuler property is "01:00" which indicates that the major rulers are shown from hour to hour. You can use the MajorTimeRuler property to programmatically change the major ruler occurrence.

The <u>DayStartTime</u> property defines the starting time of the day, and the <u>DayEndTime</u> property defines the ending time of the day. The <u>MajorTimeLabel</u> property defines the time label to be shown on the major rulers, while the <u>MinorTimeLabel</u> property specifies the time label to be shown on minor rulers. The <u>MajorLabelColor</u> property indicates the color to show the labels on major rulers. The <u>MinorTimeRuler</u> property indicates the time to show the next minor label/line.

property TimeScale.MaxWidth as Long

Gets or sets a value that indicates the maximum width for the current TimeScale object.

Туре	Description
Long	A Long expression that specifies the maximum width for the current TimeScale object

By default, the MaxWidth property is 96 pixels. The <u>MinWidth</u> property indicates the minimum width for the time scale, and the MaxWidth indicates the maximum size of the time scale. The <u>Width</u> property of the time scale indicates the width in pixels of the time scale. The <u>Visible</u> property shows or hides the time scale. The <u>AllowResize</u> property may specify whether a time scale is resizable or not, while AllowResizeTimeScale property on exDisallow may specify that no time scale can be resized at runtime.

The <u>AllowResizeTimeScale</u> property indicates the keys the user can resize at runtime the time scale. The AllowResize property may specify whether a time scale is resizable or not, while AllowResizeTimeScale property on exDisallow may specify that no time scale can be resized at runtime. The <u>LayoutStartChanging</u>(exScheduleResizeTimeScale) and <u>LayoutEndChanging</u>(exScheduleResizeTimeScale) properties notifies your application once the user resizes a time scale.

property TimeScale.MinorLabelColor as Color

Specifies the foreground color to display the labels of minor rulers.

Туре	Description
Color	A Color expression that specifies the color to show the labels on the minor rulers.

The MinorLabelColor property indicates the color to show the labels on minor rulers. The <u>MajorTimeLabel</u> property defines the time label to be shown on the major rulers, while the MinorTimeLabel property specifies the time label to be shown on minor rulers. The <u>MinorTimeLabelPlainText</u> property should be set on False, if the MinorTimeLabel property uses UI font attributes like , <fgcolor>, and so on. The <u>MinorTimeRuler</u> property indicates the time to show the minor rulers.

property TimeScale.MinorTimeLabel as String

Indicates the label to be displayed on the minor ruler of the current TimeScale object.

Туре	Description
String	A string expression that defines the extended HTML format to display the labels on the minor rulers.

By default, the MinorTimeLabel property is ":<%nn%>", which indicates that the major rulers displays the minutes only. The <u>MajorTimeLabel</u> property defines the time label to be shown on the major rulers, while the MinorTimeLabel property specifies the time label to be shown on minor rulers. The <u>MinorLabelColor</u> property indicates the color to show the labels on minor rulers. The <u>MinorTimeLabelPlainText</u> property should be set on False, if the MinorTimeLabel property uses UI font attributes like , <fgcolor>, and so on. The <u>MinorTimeRuler</u> property indicates the time to show the minor rulers.

The <u>TimeZone</u> property can be used to programmatically update the time zone. You can use the <u>ShortTimeFormat</u> property to define a 24-hours format, by removing the <%AM.PM%> TAG. The <u>Caption</u> property specifies the HTML caption to be displayed on the top side of the time scale.

The minor lines can be changed using the following **<u>Background</u>** properties:

- The Background(exScheduleTimeScaleMinorRulerColor) specifies the color to show the lines for minor rulers (This option changes the lines being shown in the time scale header)
- The Background(exScheduleTimeScaleMinorRulerStyle) property indicates a <u>LinesStyleEnum</u> expression that determines the style of lines to be shown on minor rulers (This option changes the lines being shown in the time scale header)
- The Background(exScheduleMinorTimeScaleStyle) indicates a <u>LinesStyleEnum</u> expression that determines style of lines to be shown on the schedule view.
- The Background(exScheduleMinorTimeRulerColor) indicates the color to show the minor lines on the schedule view

Here's a few samples on how you can use the MinorTimeLabel property:

- "", no label is shown on the minor rulers
- "<%hh%>:<%nn%> <%AM/PM%>", hour and minute of the minor, using the AM/PM time indicators.
- ":<%nn%>", minute with a smaller font. The <u>MinorTimeLabelPlainText</u> property must be set on False.

The MinorTimeLabel property supports the following HTML tags:

- <%hy%> Date displayed as the half of the year (1 to 2).
- <%loc_gg%> Indicates period/era using the current user regional and language settings.
- <%loc_sdate%> Indicates the date in the short format using the current user regional and language settings.
- <%loc_ldate%> Indicates the date in the long format using the current user regional and language settings.
- <%loc_dsep%> Indicates the date separator using the current user regional and language settings (/).
- <%h%> Hour in one or two digits, as needed (0 to 23).
- <%**hh**%> Hour in two digits (00 to 23).
- <%**n**%> Minute in one or two digits, as needed (0 to 59).
- <%**nn**%> Minute in two digits (00 to 59).
- <%s%> Second in one or two digits, as needed (0 to 59).
- <%**ss**%> Second in two digits (00 to 59).
- <%AM/PM%> Twelve-hour clock with the uppercase letters "AM" or "PM", as appropriate. (Use the <u>AMPM</u> property to specify the name of the AM and PM indicators). You can use the <%loc_AM/PM%> that indicates the time marker such as AM or PM using the current user regional and language settings. You can use <%loc_A/P%> that indicates the one character time marker such as A or P using the current user regional and language settings.
- <%loc_AM/PM%> Indicates the time marker such as AM or PM using the current user regional and language settings.
- <%loc_A/P%> Indicates the one character time marker such as A or P using the current user regional and language settings.
- <%loc_time%> Indicates the time using the current user regional and language settings.
- <%loc_time24%> Indicates the time in 24 hours format without a time marker using the current user regional and language settings.
- <%loc_tsep%> indicates the time separator using the current user regional and language settings (:).

The MinorTimeLabel property supports the following HTML tags (The <u>MinorTimeLabelPlainText</u> property must be set on False, else the following UI attributes are not shown) :

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

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

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

- exp, stores the plain text to be shown once the user clicks the anchor, such as "<a ;exp=show lines>"
- e64, encodes in BASE64 the HTML text to be shown once the user clicks the anchor, such as "<a/li>

;e64=gA8ABmABnABjABvABshIAOQAEAAHAAGESikWio+ABzABohp3iELABpABu " that displays show lines- in gray when the user clicks the + anchor. The "gA8ABmABnABjABvABshIAOQAEAAHAAGESikWio+ABzABohp3iELABpABuABljY string encodes the "<fgcolor 808080>show lines<a>-</fgcolor>" The Decode64Text/Encode64Text methods of the eXPrint can be used to decode/encode e64 fields.

Any ex-HTML caption can be transformed to an expandable-caption, by inserting the anchor ex-HTML tag. For instance, "<solidline>Header</solidline>

Line1<r><a ;exp=show lines>+
Line2
Line3" shows the Header in underlined and bold on the first line and Line1, Line2, Line3 on the rest. The "show lines" is shown instead of Line1, Line2, Line3 once the user clicks the + sign.

- ... displays portions of text with a different font and/or different size. For instance, the "bit" draws the bit text using the Tahoma font, on size 12 pt. If the name of the font is missing, and instead size is present, the current font is used with a different size. For instance, "bit" displays the bit text using the current font, but with a different size.
- **<fgcolor rrggbb>** ... **</fgcolor>** or **<**fgcolor=rrggbb> ... **</**fgcolor> displays text with a specified foreground color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- <bgcolor rrggbb> ... </bgcolor> or <bgcolor=rrggbb> ... </bgcolor> displays text with a specified background color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- <solidline rrggbb> ... </solidline> or <solidline=rrggbb> ... </solidline> draws a solidline 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
- number[:width] 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.
- key[:width] 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 FFFFF;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:



<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 TimeScale.MinorTimeLabelPlainText as Boolean

Specifies whether the minor label is a plain text or a formatted HTML text.

Туре	Description
Boolean	A Boolean expression that specifies whether the label on minor rulers UI font attributes like , <fgcolor> are hidden or shown.</fgcolor>

By default, the MinorTimeLabelPlainText property is True, which indicates that no UI font attributes like , <fgcolor> are displayed when any of them is included in the <u>MinorTimeLabel</u> property. The <u>MajorTimeLabel</u> property defines the time label to be shown on the major rulers, while the <u>MinorTimeLabel</u> property specifies the time label to be shown on minor rulers. Using this property on False, may affect the control's performances. Do not set on False, unless you are using any of the specified UI font tags.

property TimeScale.MinorTimeRuler as String

Indicates the minor increment for the current time scale.

Туре	Description
String	A String expression that specifies the time to show the next minor ruler.

By default, the MinorTimeRuler property is "00:15" which indicates that the minor rulers are shown from 15 to 15 minutes. You can use the MinorTimeRuler property to programmatically change the minor ruler occurrence.

The <u>DayStartTime</u> property defines the starting time of the day, and the <u>DayEndTime</u> property defines the ending time of the day. The <u>MajorTimeLabel</u> property defines the time label to be shown on the major rulers, while the MinorTimeLabel property specifies the time label to be shown on minor rulers. The <u>MinorLabelColor</u> property indicates the color to show the labels on minor rulers. The <u>MajorTimeRuler</u> property indicates the time to show the next major label/line.

property TimeScale.MinWidth as Long

Gets or sets a value that indicates the minimum width for the current TimeScale object.

Туре	Description
Long	A Long expression that specifies the minimum width for the current TimeScale object

By default, the MinWidth property is 0. The MinWidth property indicates the minimum width for the time scale, and the <u>MaxWidth</u> indicates the maximum size of the time scale. The <u>Width</u> property of the time scale indicates the width in pixels of the time scale. The <u>Visible</u> property shows or hides the time scale. The <u>AllowResize</u> property may specify whether a time scale is resizable or not, while AllowResizeTimeScale property on exDisallow may specify that no time scale can be resized at runtime.

The <u>AllowResizeTimeScale</u> property indicates the keys the user can resize at runtime the time scale. The AllowResize property may specify whether a time scale is resizable or not, while AllowResizeTimeScale property on exDisallow may specify that no time scale can be resized at runtime. The <u>LayoutStartChanging</u>(exScheduleResizeTimeScale) and <u>LayoutEndChanging</u>(exScheduleResizeTimeScale) properties notifies your application once the user resizes a time scale.
property TimeScale.Position as Long

Gets or sets the position of the current time scale.

Туре	Description
Long	A Long expression that indicates the position of the time scale.

By default, the Position property is 0, which indicates that the time scale is left displayed on the first position. The Position property indicates the position of the time scale as they are displayed. The <u>AlignLeft</u> property can be used to programmatically change the time scale alignment. The <u>AllowMoveTimeScale</u> property indicates the keys the user can move at runtime the time scale from a side to another. The <u>AlignLeft</u> and Position properties may be changed if the user moves the time scale position to a new side, while the <u>AllowMoveTimeScale</u> property is not exDisallow (0).

property TimeScale.RulerBackColor as Color

Specifies the background color for TimeScale's ruler.

Туре	Description
Color	A Color expression that specifies the background color to show the ruler part of the time scale. The last 7 bits in the high significant byte of the color to indicates the identifier of the skin being used. Use the <u>Add</u> 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 RulerBackColor property is initialized with the value of the <u>Background</u>(exScheduleTimeScaleRulerBackColor) property. By default, the <u>Background</u>(exScheduleTimeScaleRulerBackColor) property defines the background color to show the ruler part of the control's time scales. If the RulerBackColor property is zero (default), the <u>BackColor</u> property indicates the ruler's background color.

The time scale supports the following background properties:

- The <u>BackColor</u> property changes the time scale's background.
- The <u>CaptionBackColor</u> property defines the color to show caption of the timescale. The <u>Caption</u> property defines the HTML caption to be displayed on the top of the time scale.
- The RulerBackColor property specifies the background color for the rulers part of the time scale. The <u>MajorTimeLabel</u> property defines the time label to be shown on the major rulers, while the <u>MinorTimeLabel</u> property specifies the time label to be shown on minor rulers.

All background colors can display a solid color as well as an EBN object.

The following sample shows the time scale using different background colors:



property TimeScale.TimeZone as String

Indicates the time zone for the current time scale.

Туре	Description
String	A String expression that defines the time-zone to be displayed. In other words, the time to be added to current time. For instance, the "+03:00" adds a three hours to the current time, while "-02:00" delays the current time scale with 2 hours earlier.

The TimeZone parameter of the <u>Add</u> method initializes the TimeZone property. The TimeZone property can be used to programmatically update the time zone. The <u>Caption</u> property specifies the HTML caption to be displayed on the top side of the time scale. The <u>MajorTimeRuler</u> property indicates the time to increment the major rulers, while the <u>MinorTimeRuler</u> property specifies the time to increment the minor rulers.

The <u>DayStartTime</u> property defines the starting time of the day, and the <u>DayEndTime</u> property defines the ending time of the day.

property TimeScale.ToolTip as String

Indicates the tooltip of the TimeScale object.

Туре	Description
String	A String expression that specifies the extended HTML caption to be displayed when cursor hovers the time scale.

By default, the ToolTip property is "". The <u>Caption</u> property specifies the HTML caption to be displayed on the top side of the time scale. The <u>MajorTimeLabel</u> property defines the time label to be shown on the major rulers, while the <u>MinorTimeLabel</u> property specifies the time label to be shown on minor rulers. The <u>TimeZone</u> property can be used to programmatically update the time zone.

Here's a few samples on how you can use the ToolTip property:

- "this is a bit of text that's being displayed when cursor hovers the time scale", an hard coded text
- "<%hh%>:<%nn%>", hour and minute from the cursor, in the 24-hours format
- "Time: <%loc_time24%>", hour and minute from the cursor, in the 24-hours format, using the current regional settings
- "Time
<%loc_time%>" displays the Time text on the first line, while on the second line is shows the time from cursor

The ToolTip property supports the following HTML tags:

- <%hy%> Date displayed as the half of the year (1 to 2).
- <%loc_gg%> Indicates period/era using the current user regional and language settings.
- <%loc_sdate%> Indicates the date in the short format using the current user regional and language settings.
- <%loc_ldate%> Indicates the date in the long format using the current user regional and language settings.
- <%loc_dsep%> Indicates the date separator using the current user regional and language settings (/).
- <%h%> Hour in one or two digits, as needed (0 to 23).
- <%**hh**%> Hour in two digits (00 to 23).
- <%**n**%> Minute in one or two digits, as needed (0 to 59).
- <%nn%> Minute in two digits (00 to 59).
- <%**s**%> Second in one or two digits, as needed (0 to 59).
- <%**ss**%> Second in two digits (00 to 59).
- <%AM/PM%> Twelve-hour clock with the uppercase letters "AM" or "PM", as

appropriate. (Use the <u>AMPM</u> property to specify the name of the AM and PM indicators). You can use the <%loc_AM/PM%> that indicates the time marker such as AM or PM using the current user regional and language settings. You can use <%loc_A/P%> that indicates the one character time marker such as A or P using the current user regional and language settings

- <%loc_AM/PM%> Indicates the time marker such as AM or PM using the current user regional and language settings.
- <%loc_A/P%> Indicates the one character time marker such as A or P using the current user regional and language settings.
- <%loc_time%> Indicates the time using the current user regional and language settings.
- <%loc_time24%> Indicates the time in 24 hours format without a time marker using the current user regional and language settings.
- <%loc_tsep%> indicates the time separator using the current user regional and language settings (:).

The ToolTip property supports the following HTML tags:

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

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

- exp, stores the plain text to be shown once the user clicks the anchor, such as "<a ;exp=show lines>"
- e64, encodes in BASE64 the HTML text to be shown once the user clicks the anchor, such as "<a/li>

;e64=gA8ABmABnABjABvABshIAOQAEAAHAAGESikWio+ABzABohp3iELABpABu " that displays show lines- in gray when the user clicks the + anchor. The "gA8ABmABnABjABvABshIAOQAEAAHAAGESikWio+ABzABohp3iELABpABuABljY string encodes the "<fgcolor 808080>show lines<a>-</fgcolor>" The Decode64Text/Encode64Text methods of the eXPrint can be used to decode/encode e64 fields.

Any ex-HTML caption can be transformed to an expandable-caption, by inserting the anchor ex-HTML tag. For instance, "<solidline>Header</solidline>

Line1<r><a ;exp=show lines>+
Line2
Line3" shows the Header in underlined and bold on the first line and Line1, Line2, Line3 on the rest. The "show lines" is shown instead of Line1, Line2, Line3 once the user clicks the + sign.

- ... displays portions of text with a different font and/or different size. For instance, the "bit" draws the bit text using the Tahoma font, on size 12 pt. If the name of the font is missing, and instead size is present, the current font is used with a different size. For instance, "bit" displays the bit text using the current font, but with a different size.
- **<fgcolor rrggbb>** ... **</fgcolor>** or **<**fgcolor=rrggbb> ... **</**fgcolor> displays text with a specified foreground color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- <bgcolor rrggbb> ... </bgcolor> or <bgcolor=rrggbb> ... </bgcolor> displays text with a specified background color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- <solidline rrggbb> ... </solidline> or <solidline=rrggbb> ... </solidline> draws a solid-line on the bottom side of the current text-line, of specified RGB color. The <solidline> ... </solidline> draws a black solid-line on the bottom side of the current text-line. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- <dotline rrggbb> ... </dotline> or <dotline=rrggbb> ... </dotline> draws a dot-line on the bottom side of the current text-line, of specified RGB color. The <dotline> ...
 </dotline> draws a black dot-line on the bottom side of the current text-line. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<upline>** ... **</upline>** draws the line on the top side of the current text-line (requires <solidline> or <dotline>).
- <r> right aligns the text
- <c> centers the text
-
 forces a line-break
- number[:width] 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.
- key[:width] 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 FFFFF;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 TimeScale.UserData as Variant

Indicates any extra data associated with the TimeScale object.

Туре	Description
Variant	A VARIANT expression that indicates any extra data you can associate with the time scale.

By default, the UserData property is empty. You can use the UserData property to associate any extra data to the current time scale. The <u>Caption</u> property defines the HTML caption to be displayed on the top of the time scale. The <u>MajorTimeLabel</u> property defines the time label to be shown on the major rulers, while the <u>MinorTimeLabel</u> property specifies the time label to be shown on minor rulers.

property TimeScale.Visible as Boolean

Specifies whether the TimeScale is visible or hidden.

Туре	Description
Boolean	A Boolean expression that specifies whether the time scale is visible or hidden.

By default, the Visible property is True, which means that the time scale is shown. The Visible property shows or hides the time scale. The <u>Position</u> property indicates the position of the time scale. The <u>Width</u> property of the time scale indicates the width in pixels of the time scale. The <u>MinWidth</u> property indicates the minimum width for the time scale, and the <u>MaxWidth</u> indicates the maximum size of the time scale. The <u>AllowResize</u> property may specify whether a time scale is resizable or not, while AllowResizeTimeScale property on exDisallow may specify that no time scale can be resized at runtime.

The <u>AllowResizeTimeScale</u> property indicates the keys the user can resize at runtime the time scale. The AllowResize property may specify whether a time scale is resizable or not, while AllowResizeTimeScale property on exDisallow may specify that no time scale can be resized at runtime. The <u>LayoutStartChanging</u>(exScheduleResizeTimeScale) and <u>LayoutEndChanging</u>(exScheduleResizeTimeScale) properties notifies your application once the user resizes a time scale.

property TimeScale.Width as Long

Gets or sets a value that indicates the TimeScale's width, in pixels.

Туре	Description
Long	A Long expression that specifies the width of the time scale.

By default, the Width property is 64 pixels. The Width property of the time scale indicates the width in pixels of the time scale. The <u>MinWidth</u> property indicates the minimum width for the time scale, and the <u>MaxWidth</u> indicates the maximum size of the time scale. The <u>Visible</u> property shows or hides the time scale. The <u>AllowResize</u> property may specify whether a time scale is resizable or not, while AllowResizeTimeScale property on exDisallow may specify that no time scale can be resized at runtime.

The <u>AllowResizeTimeScale</u> property indicates the keys the user can resize at runtime the time scale. The AllowResize property may specify whether a time scale is resizable or not, while AllowResizeTimeScale property on exDisallow may specify that no time scale can be resized at runtime. The <u>LayoutStartChanging</u>(exScheduleResizeTimeScale) and <u>LayoutEndChanging</u>(exScheduleResizeTimeScale) properties notifies your application once the user resizes a time scale.

TimeScales object

The TimeScales object holds a collection of <u>TimeScale</u> objects. The control handles one or more time scales. Each time scale can display a different time zone, and can be aligned to any side of the schedule view. The TimeScales collection is accessible through the <u>TimeScales</u> property of the control. The <u>TimeScaleFromPoint</u> method gets the TimeScale object from the cursor. The <u>TimeScalesFont</u> defines the font to display the control's time scales. The <u>TimeZone</u> property defines the time zone of the time scale. The <u>AllowMoveTimeScale</u> property indicates the keys the user can move at runtime the time scale from a side to another. The <u>AllowResizeTimeScale</u> property indicates the keys the user can move at runtime the scale user can resize at runtime the time scale. The <u>DayStartTime</u> property defines the starting time of the day, and the <u>DayEndTime</u> property defines the ending time of the day.

The following screen show shows the control with two time-scales:



The TimeScales object support the following properties and methods:

Name	Description
Add	Adds a TimeScale object to the collection and returns a reference to the newly created object.
<u>Clear</u>	Removes all objects in a collection.
<u>Count</u>	Returns the number of objects in a collection.

<u>ltem</u>	Returns a specific TimeScale of the TimeScales collection.
<u>Remove</u>	Removes a specific member from the TimeScales collection.

method TimeScales.Add (TimeZone as String)

Adds a TimeScale object to the collection and returns a reference to the newly created object.

Туре	Description
TimeZone as String	A String expression that defines the time-zone to be displayed. In other words, the time to be added to current time. The <u>TimeZone</u> property can be used to programmatically update the time zone. For instance, the "+03:00" adds a three hours to the current time, while "-02:00" delays the current time scale with 2 hours earlier.
Return	Description
<u>TimeScale</u>	A TimeScale object being created.

By default, the control adds a TimeScale object that can be accessed through the TimeScales(0) property. The newly added time scale is aligned to left, so you can use the <u>AlignLeft</u> property on False, to align the time scale to the right. The <u>TimeZone</u> property can be used to programmatically update the time zone. The <u>Visible</u> property shows or hides the time scale. The <u>Position</u> property changes the position of the time scale from left to right.

The <u>DayStartTime</u> property defines the starting time of the day, and the <u>DayEndTime</u> property defines the ending time of the day. The <u>TimeScaleFromPoint</u> method gets the TimeScale object from the cursor. The <u>AllowResizeTimeScale</u> property indicates the keys the user can resize at runtime the time scale.

method TimeScales.Clear ()

Removes all objects in a collection.

Туре

Description

The Clear method removes all time-scales from the control. The <u>Remove</u> method removes a time scale giving its time zone or its index. The <u>Visible</u> property shows or hides the time scale. The <u>Position</u> property changes the position of the time scale from left to right. The <u>Item</u> property accesses the time scale object its time zone or its index. The <u>TimeZone</u> property can be used to programmatically update the time zone. The <u>Index</u> property defines the index of the time scale in the TimeScales collection.

property TimeScales.Count as Long

Returns the number of objects in a collection.

Туре	Description
Long	A long expression that specifies the number of time scales in the control.

The Count property indicates the number of time scales in the control. The <u>Item</u> property accesses the time scale object its time zone or its index. The <u>Remove</u> method removes a time scale giving its time zone or its index. The <u>Clear</u> method removes all time-scales from the control. The <u>Visible</u> property shows or hides the time scale. The <u>Position</u> property changes the position of the time scale from left to right. The <u>TimeZone</u> property can be used to programmatically update the time zone. The <u>Index</u> property defines the index of the time scale in the TimeScales collection. The **for each statement** is supported by the TimeScales collection, so the TimeScales collection can be enumerated using a sample like *for each ts in TimeScales*

property TimeScales.Item (Index as Variant) as TimeScale

Returns a specific TimeScale of the TimeScales collection.

Туре	Description
Index as Variant	A long expression that specifies the index of the time scale to be accessed, or a string expression that indicates the time-zone of the time scale to be accessed.
<u>TimeScale</u>	A TimeScale object associated with giving index or time- zone, or empty/nothing/NULL if no object is associated.

The Item property accesses the time scale object its time zone or its index. The <u>Count</u> property indicates the number of time scales in the control. The <u>Remove</u> method removes a time scale giving its time zone or its index. The <u>Clear</u> method removes all time-scales from the control. The <u>Visible</u> property shows or hides the time scale. The <u>Position</u> property changes the position of the time scale from left to right. The <u>TimeZone</u> property can be used to programmatically update the time zone. The <u>Index</u> property defines the index of the time scale in the TimeScales collection. The <u>TimeScaleFromPoint</u> method gets the TimeScale object from the cursor. The **for each statement** is supported by the TimeScales collection, so the TimeScales collection can be enumerated using a sample like *for each ts in TimeScales*

method TimeScales.Remove (Index as Variant)

Removes a specific member from the TimeScales collection.

Туре	Description
Index as Variant	A long expression that specifies the index of the time scale to be removed, or a string expression that indicates the time-zone of the time scale to be removed.

The Remove method removes a time scale giving its time zone or its index. The <u>Clear</u> method removes all time-scales from the control. The <u>Visible</u> property shows or hides the time scale. The <u>Position</u> property changes the position of the time scale from left to right. The <u>Item</u> property accesses the time scale object its time zone or its index. The <u>TimeZone</u> property can be used to programmatically update the time zone. The <u>Index</u> property defines the index of the time scale in the TimeScales collection.

ExSchedule events

The ExSchedule component supports the following events:

Name	Description
<u>AddEvent</u>	Notifies your application once the a new event is added.
AnchorClick	Occurs when an anchor element is clicked.
<u>ChangeEvent</u>	Occurs once an event is added, removed or updated/changed.
<u>Click</u>	Occurs when the user presses and then releases the left mouse button over the control.
<u>DblClick</u>	Occurs when the user dblclk the left mouse button over an object.
<u>Error</u>	Fired when an internal error occurs.
<u>Event</u>	Notifies the application once the control fires an event.
<u>KeyDown</u>	Occurs when the user presses a key while an object has the focus.
<u>KeyPress</u>	Occurs when the user presses and releases an ANSI key.
<u>KeyUp</u>	Occurs when the user releases a key while an object has the focus.
LayoutEndChanging	Notifies your application once the control's layout has been changed.
LayoutStartChanging	Occurs when the control's layout is about to be changed.
<u>MouseDown</u>	Occurs when the user presses a mouse button.
MouseMove	Occurs when the user moves the mouse.
<u>MouseUp</u>	Occurs when the user releases a mouse button.
OLECompleteDrag	Occurs when a source component is dropped onto a target component, informing the source component that a drag action was either performed or canceled
<u>OLEDragDrop</u>	Occurs when a source component is dropped onto a target component when the source component determines that a drop can occur.
<u>OLEDragOver</u>	Occurs when one component is dragged over another.
OLEGiveFeedback	Allows the drag source to specify the type of OLE drag- and-drop operation and the visual feedback.
	Occurs on a drag source when a drop target calls the

<u>OLESetData</u>	GetData method and there is no data in a specified format in the OLE drag-and-drop DataObject.
<u>OLEStartDrag</u>	Occurs when the OLEDrag method is called.
PictureClick	Occurs when the user clicks a picture within an event (Event.Pictures/ExtraPictures).
<u>RClick</u>	Occurs once the user right clicks the control.
<u>RemoveEvent</u>	Occurs once an event is removed.
ScrollButtonClick	Occurs when the user clicks a button in the scrollbar.
<u>UpdateEvent</u>	Notifies your application once the event changes the starting or ending margins.

event AddEvent (Ev as Event)

Notifies your application once the a new event is added.

Туре	Description
Ev as <u>Event</u>	The Ev object indicates the new appointment/event being added.

The control fires the AddEvent event once the user creates a new appointment, or the <u>Events.Add</u> method has been invoked. You can use the AddEvent event to change the event's label, pictures, and so on. You can use the <u>UserData</u> property to associate any extra data to your event/appointment. The AddEvent event is fired also during loading an XML document using the LoadXML method. The control fires the LayoutStartChanging(*exScheduleCreateEvent*)/LayoutEndChanging(*exScheduleCreateEvent*) event once the user creates a new event using the mouse. The <u>Start</u> and <u>End</u> properties of the Event are known at the moment the AddEvent event occurs. You can handle the AddEvent event to invoke your dialogs in other to update any other property of the newly added event. You can also, call the <u>Remove</u> event in case you need to remove the event. The <u>ChangeEvent(exAddEvent</u>) event is equivalent with the AddEvent event.

The <u>AllowCreateEvent</u> property specifies the keys combination to let user creates new events at runtime. The <u>CreateEventLabel</u> property indicates the label to be shown when the user creates a new event at runtime. You can use the <u>Background</u>(exScheduleCreateEventBackColor) and <u>Background</u>(exScheduleCreateEventForeColor) properties to specify the visual appearance of the events being updated at runtime * moving or resizing).

Syntax for AddEvent event, /NET version, on:

c# private void AddEvent(object sender,exontrol.EXSCHEDULELib.Event Ev)
{

VB

Private Sub AddEvent(ByVal sender As System.Object,ByVal Ev As exontrol.EXSCHEDULELib.Event) Handles AddEvent End Sub

Syntax for AddEvent event, /COM version, on:

private void AddEvent(object sender,
 AxEXSCHEDULELib._IScheduleEvents_AddEventEvent e)

	{ }
C++	void OnAddEvent(LPDISPATCH Ev) { }
C++ Builder	voidfastcall AddEvent(TObject *Sender,Exschedulelib_tlb::IEvent *Ev) { }
Delphi	procedure AddEvent(ASender: TObject; Ev : IEvent); begin end;
Delphi 8 (.NET only)	procedure AddEvent(sender: System.Object; e: AxEXSCHEDULELibIScheduleEvents_AddEventEvent); begin end;
Powe	begin event AddEvent(oleobject Ev) end event AddEvent
VB.NET	Private Sub AddEvent(ByVal sender As System.Object, ByVal e As AxEXSCHEDULELibIScheduleEvents_AddEventEvent) Handles AddEvent End Sub
VB6	Private Sub AddEvent(ByVal Ev As EXSCHEDULELibCtl.IEvent) End Sub
VBA	Private Sub AddEvent(ByVal Ev As Object) End Sub
VFP	LPARAMETERS Ev
Xbas	PROCEDURE OnAddEvent(oSchedule,Ev)

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

Java	<script event="AddEvent(Ev)" language="JScript"> </script>
VBSc	<script language="VBScript"> Function AddEvent(Ev) End Function </script>
Visual Data…	Procedure OnComAddEvent Variant IIEv Forward Send OnComAddEvent IIEv End_Procedure
Visual Objects	METHOD OCX_AddEvent(Ev) CLASS MainDialog RETURN NIL
X++	void onEvent_AddEvent(COM _Ev) { }
XBasic	function AddEvent as v (Ev as OLE::Exontrol.Schedule.1::IEvent) end function
dBASE	function nativeObject_AddEvent(Ev) return

The following VB sample shows how you can change the event's pattern once a new event is created:

```
Private Sub Schedule1_AddEvent(ByVal Ev As EXSCHEDULELibCtl.IEvent)
With Ev.BodyPattern
.Type = exPatternBDiagonal
.Color = RGB(128, 128, 128)
End With
End Sub
```

The following VB sample asks the user if he wants to keep the newly created event, and if

not, removes it:

```
Dim iCreatingEvent As Long
Private Sub Schedule1_AddEvent(ByVal Ev As EXSCHEDULELibCtl.IEvent)
  If Not (iCreatingEvent = 0) Then
    If Not MsgBox("Do you allow creating this new event?", vbQuestion Or
vbYesNoCancel) = vbYes Then
      Schedule1.Events.Remove (Ev.Handle)
    End If
  End If
End Sub
Private Sub Schedule1_LayoutStartChanging(ByVal Operation As
EXSCHEDULELibCtl.LayoutChangingEnum)
  If (Operation = exScheduleCreateEvent) Then
    iCreatingEvent = iCreatingEvent + 1
  End If
End Sub
Private Sub Schedule1_LayoutEndChanging(ByVal Operation As
EXSCHEDULELibCtl.LayoutChangingEnum)
  If (Operation = exScheduleCreateEvent) Then
    iCreatingEvent = iCreatingEvent - 1
  End If
End Sub
```

event AnchorClick (AnchorID as String, Options as String)

Occurs when an anchor element is clicked.

Туре	Description
AnchorID as String	A string expression that specifies the identifier of the anchor being clicked. For instance if you have a <a id;options="">anchor , the id is the AnchorID, while the options is passed to the Options parameter.
Options as String	A string expression that specifies the options being used in the <a ;options=""> declaration.

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

Currently, the recognizable anchors elements are in the event's body, by using the $\langle a \rangle$ elements in the <u>LongLabel</u> or <u>ExtraLabel</u> property.

Syntax for AnchorClick event, /NET version, on:

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

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

Syntax for AnchorClick event, /COM version, on:

C#	private void AnchorClick(object sender, AxEXSCHEDULELibIScheduleEvents_AnchorClickEvent e) { }
C++	void OnAnchorClick(LPCTSTR AnchorID,LPCTSTR Options) { }
C++ Builder	<pre>voidfastcall AnchorClick(TObject *Sender,BSTR AnchorID,BSTR Options) { }</pre>
Delphi	procedure AnchorClick(ASender: TObject; AnchorID : WideString;Options : WideString); begin end;
Delphi 8 (.NET only)	procedure AnchorClick(sender: System.Object; e: AxEXSCHEDULELibIScheduleEvents_AnchorClickEvent); begin end;
Powe	begin event AnchorClick(string AnchorID,string Options) end event AnchorClick
VB.NET	Private Sub AnchorClick(ByVal sender As System.Object, ByVal e As AxEXSCHEDULELibIScheduleEvents_AnchorClickEvent) Handles AnchorClick End Sub
VB6	Private Sub AnchorClick(ByVal AnchorID As String,ByVal Options As String) End Sub
VBA	Private Sub AnchorClick(ByVal AnchorID As String,ByVal Options As String) End Sub
VFP	LPARAMETERS AnchorID,Options

Xbas...

Syntax f	or AnchorClick event, /COM version ^(others) , on:
Java	<script event="AnchorClick(AnchorID,Options)" language="JScript"> </script>
VBSc	<script language="VBScript"> Function AnchorClick(AnchorID,Options) End Function </script>
Visual Data	Procedure OnComAnchorClick String IIAnchorID String IIOptions Forward Send OnComAnchorClick IIAnchorID IIOptions End_Procedure
Visual Objects	METHOD OCX_AnchorClick(AnchorID,Options) CLASS MainDialog RETURN NIL
X++	void onEvent_AnchorClick(str _AnchorID,str _Options) { }
XBasic	function AnchorClick as v (AnchorID as C,Options as C) end function
dBASE	function nativeObject_AnchorClick(AnchorID,Options) return

The following samples adds a <a rem>remove to all events:

VBA (MS Access, Excell...)

```
With Schedule1
.DefaultEventLongLabel = "<%=%256%><br><a rem>remove</a>"
End With
```

VB6

```
With Schedule1
.DefaultEventLongLabel = "<%=%256%><br><a rem>remove</a>"
End With
```

VB.NET

```
With Exschedule1
.DefaultEventLongLabel = "<%=%256%> <br> <a rem>remove</a>"
End With
```

VB.NET for /COM

With AxSchedule1 .DefaultEventLongLabel = "<%=%256%>
 <a rem>remove" End With

C++

```
Copy and paste the following directives to your header file as
it defines the namespace 'EXSCHEDULELib' for the library: 'ExSchedule 1.0 Control
Library'
```

```
#import <ExSchedule.dll>
using namespace EXSCHEDULELib;
*/
EXSCHEDULELib::ISchedulePtr spSchedule1 = GetDlgItem(IDC_SCHEDULE1)-
>GetControlUnknown();
spSchedule1->PutDefaultEventLongLabel(L"<%=%256%><br><a
rem>remove</a>");
```

C++ Builder

Schedule1->DefaultEventLongLabel = L"<%=%256%>
<a rem>remove";

```
exschedule1.DefaultEventLongLabel = "<%=%256%> <br> <a rem>remove</a>";
```

JavaScript

```
<OBJECT classid="clsid:9B09E13D-7A88-4299-9DBE-383380435377"
id="Schedule1"></OBJECT>
```

```
<SCRIPT LANGUAGE="JScript">
```

Schedule1.DefaultEventLongLabel = "<%=%256%>
 <a rem>remove"; </SCRIPT>

C# for /COM

```
axSchedule1.DefaultEventLongLabel = "<%=%256%> <br> <a rem>remove</a>";
```

X++ (Dynamics Ax 2009)

```
public void init()
{
;
super();
exschedule1.DefaultEventLongLabel("<%=%256%> <br> <a rem>remove</a>");
}
```

Delphi 8 (.NET only)

```
with AxSchedule1 do
begin
DefaultEventLongLabel := '<%=%256%> <br> <a rem>remove</a>';
end
```

Delphi (standard)

```
with Schedule1 do begin
```

```
DefaultEventLongLabel := '<%=%256%> <br> <a rem>remove</a>';
end
```

VFP

```
with thisform.Schedule1
   .DefaultEventLongLabel = "<%=%256%> <br> <a rem>remove</a>"
endwith
```

dBASE Plus

local oSchedule oSchedule = form.Activex1.nativeObject oSchedule.DefaultEventLongLabel = "<%=%256%>
 <a rem> remove"

XBasic (Alpha Five)

```
Dim oSchedule as P
oSchedule = topparent:CONTROL_ACTIVEX1.activex
oSchedule.DefaultEventLongLabel = "<%=%256%> <br> <a rem>remove</a>"
```

Visual Objects

```
oDCOCX_Exontrol1:DefaultEventLongLabel := "<%=%256%> <br> <a
rem>remove</a>"
```

PowerBuilder

```
OleObject oSchedule
oSchedule = ole_1.Object
oSchedule.DefaultEventLongLabel = "<%=%256%> <br> <a rem>remove</a>"
```

The following samples displays the identifier of the anchor/<a> being clicked.

AnchorClick event - Occurs when an anchor element is clicked.

Private Sub Schedule1_AnchorClick(ByVal AnchorID As String,ByVal Options As String) With Schedule1 Debug.Print(AnchorID) End With End Sub

VB6

' **AnchorClick event - Occurs when an anchor element is clicked.** Private Sub Schedule1_AnchorClick(ByVal AnchorID As String,ByVal Options As String)

With Schedule1 Debug.Print(AnchorID) End With End Sub

VB.NET

' AnchorClick event - Occurs when an anchor element is clicked.

Private Sub Exschedule1_AnchorClick(ByVal sender As System.Object,ByVal AnchorID As String,ByVal Options As String) Handles Exschedule1.**AnchorClick** With Exschedule1 Debug.Print(AnchorID) End With End Sub

VB.NET for /COM

' AnchorClick event - Occurs when an anchor element is clicked.

Private Sub AxSchedule1_AnchorClick(ByVal sender As System.Object, ByVal e As AxEXSCHEDULELib._IScheduleEvents_AnchorClickEvent) Handles

AxSchedule1. AnchorClick

With AxSchedule1 Debug.Print(e.anchorID) End With End Sub

C++

```
// AnchorClick event - Occurs when an anchor element is clicked.
void OnAnchorClickSchedule1(LPCTSTR AnchorID,LPCTSTR Options)
{
    /*
        Copy and paste the following directives to your header file as
        it defines the namespace 'EXSCHEDULELib' for the library: 'ExSchedule 1.0
Control Library'
        #import <ExSchedule.dll>
        using namespace EXSCHEDULELib;
        */
        EXSCHEDULELib::ISchedulePtr spSchedule1 = GetDlgItem(IDC_SCHEDULE1)-
>GetControlUnknown();
        OutputDebugStringW(L"AnchorID");
}
```

C++ Builder

```
// AnchorClick event - Occurs when an anchor element is clicked.
void __fastcall TForm1::Schedule1AnchorClick(TObject *Sender,BSTR AnchorID,BSTR
Options)
{
    OutputDebugString( L"AnchorID" );
}
```

```
// AnchorClick event - Occurs when an anchor element is clicked.
private void exschedule1_AnchorClick(object sender,string AnchorID,string Options)
{
    System.Diagnostics.Debug.Print( AnchorID.ToString() );
}
//this.exschedule1.AnchorClick += new
exontrol.EXSCHEDULELib.exg2antt.AnchorClickEventHandler(this.exschedule1_/
```

JavaScript

```
<SCRIPT FOR="Schedule1" EVENT="AnchorClick(AnchorID,Options)"
LANGUAGE="JScript">
alert(AnchorID);
</SCRIPT>
<OBJECT classid="clsid:9B09E13D-7A88-4299-9DBE-383380435377"
id="Schedule1"> </OBJECT>
<SCRIPT LANGUAGE="JScript">
</SCRIPT>
```

C# for /COM

```
// AnchorClick event - Occurs when an anchor element is clicked.
private void axSchedule1_AnchorClick(object sender,
AxEXSCHEDULELib._IScheduleEvents_AnchorClickEvent e)
{
    System.Diagnostics.Debug.Print( e.anchorID.ToString() );
}
//this.axSchedule1.AnchorClick += new
AxEXSCHEDULELib._IScheduleEvents_AnchorClickEventHandler(this.axSchedule1
```

X++ (Dynamics Ax 2009)

```
// AnchorClick event - Occurs when an anchor element is clicked.
void onEvent_AnchorClick(str _AnchorID,str _Options)
{
    ;
    print(_AnchorID);
}
public void init()
{
    ;
    super();
}
```

Delphi 8 (.NET only)



Delphi (standard)

// AnchorClick event - Occurs when an anchor element is clicked.

procedure TForm1.Schedule1AnchorClick(ASender: TObject; AnchorID : WideString;Options : WideString);

```
begin
with Schedule1 do
begin
OutputDebugString( AnchorID );
end
end;
```

VFP

*** **AnchorClick** event - Occurs when an anchor element is clicked. *** LPARAMETERS AnchorID,Options with thisform.Schedule1 DEBUGOUT(AnchorID) endwith

dBASE Plus

```
/*
with (this.ACTIVEX1.nativeObject)
    AnchorClick = class::nativeObject_AnchorClick
endwith
*/
// Occurs when an anchor element is clicked.
function nativeObject_AnchorClick(AnchorID,Options)
    local oSchedule
    oSchedule = form.Activex1.nativeObject
    ? Str(AnchorID)
return
local oSchedule
oSchedule
oSchedule = form.Activex1.nativeObject
```
XBasic (Alpha Five)

'Occurs when an anchor element is clicked.
function AnchorClick as v (AnchorID as C,Options as C)
Dim oSchedule as P
oSchedule = topparent:CONTROL_ACTIVEX1.activex
? AnchorID
end function
Dim oSchedule as P

oSchedule = topparent:CONTROL_ACTIVEX1.activex

Visual Objects

METHOD OCX_Exontrol1AnchorClick(AnchorID,Options) CLASS MainDialog // AnchorClick event - Occurs when an anchor element is clicked. OutputDebugString(String2Psz(AsString(AnchorID))) RETURN NIL

PowerBuilder

oSchedule = ole_1.Object

```
/*begin event AnchorClick(string AnchorID,string Options) - Occurs when an anchor
element is clicked.*/
/*
        OleObject oSchedule
        oSchedule = ole_1.Object
        MessageBox("Information",string( String(AnchorID) ))
*/
/*end event AnchorClick*/
OleObject oSchedule
```

event ChangeEvent (Ev as Event, Operation as ChangeOperationEnum)

Occurs once an event is added, removed or updated/changed.

Туре	Description
Ev as <u>Event</u>	An <u>Event</u> object being added, removed or changed.
Operation as <u>ChangeOperationEnum</u>	A <u>ChangeOperationEnum</u> type that specifies the operation that just occured.

The ChangeEvent event notifies your application once a new event is added, if any event is removed, or a any property of the event is changed. The ChangeEvent(exAddEvent) event is equivalent with the AddEvent event. The ChangeEvent(exRemoveEvent) event is equivalent with the <u>RemoveEvent</u> event. The ChangeEvent(exUpdateEvent + <u>EventKnownPropertyEnum</u>) event occurs once the <u>EventKnownPropertyEnum</u> property of the Ev is changed. For instance, the ChangeEvent(exUpdateEvent + exEventEndDateTime) event (16 + 2 = 18) indicates that the ending position of the event (Ev) is changed, or the end margin of the event is resized.

Syntax for ChangeEvent event, /NET version, on:

- C# private void ChangeEvent(object sender,exontrol.EXSCHEDULELib.Event Ev,exontrol.EXSCHEDULELib.ChangeOperationEnum Operation) { }
 - Private Sub ChangeEvent(ByVal sender As System.Object,ByVal Ev As exontrol.EXSCHEDULELib.Event,ByVal Operation As exontrol.EXSCHEDULELib.ChangeOperationEnum) Handles ChangeEvent End Sub

Syntax for ChangeEvent event, /COM version, on:

C# private void ChangeEvent(object sender, AxEXSCHEDULELib._IScheduleEvents_ChangeEventEvent e) { }

void OnChangeEvent(LPDISPATCH Ev,long Operation)

C++ Builder	<pre>voidfastcall ChangeEvent(TObject *Sender,Exschedulelib_tlb::IEvent *Ev,Exschedulelib_tlb::ChangeOperationEnum Operation) { }</pre>
Delphi	procedure ChangeEvent(ASender: TObject; Ev : IEvent;Operation : ChangeOperationEnum); begin end;
Delphi 8 (.NET only)	procedure ChangeEvent(sender: System.Object; e: AxEXSCHEDULELibIScheduleEvents_ChangeEventEvent); begin end;
Powe	begin event ChangeEvent(oleobject Ev,long Operation) end event ChangeEvent
VB.NET	Private Sub ChangeEvent(ByVal sender As System.Object, ByVal e As AxEXSCHEDULELibIScheduleEvents_ChangeEventEvent) Handles ChangeEvent End Sub
VB6	Private Sub ChangeEvent(ByVal Ev As EXSCHEDULELibCtl.IEvent,ByVal Operation As EXSCHEDULELibCtl.ChangeOperationEnum) End Sub
VBA	Private Sub ChangeEvent(ByVal Ev As Object,ByVal Operation As Long) End Sub
VFP	LPARAMETERS Ev,Operation
Xbas	PROCEDURE OnChangeEvent(oSchedule,Ev,Operation) RETURN

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

Java... <SCRIPT EVENT="ChangeEvent(Ev,Operation)" LANGUAGE="JScript">

l	
VBSc	<script language="VBScript"> Function ChangeEvent(Ev,Operation) End Function </script>
Visual Data	Procedure OnComChangeEvent Variant IIEv OLEChangeOperationEnum IIOperation Forward Send OnComChangeEvent IIEv IIOperation End_Procedure
Visual Objects	METHOD OCX_ChangeEvent(Ev,Operation) CLASS MainDialog RETURN NIL
X++	<pre>void onEvent_ChangeEvent(COM _Ev,int _Operation) { }</pre>
XBasic	function ChangeEvent as v (Ev as OLE::Exontrol.Schedule.1::IEvent,Operation as OLE::Exontrol.Schedule.1::ChangeOperationEnum) end function
dBASE	function nativeObject_ChangeEvent(Ev,Operation) return

event Click ()

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

Туре

Description

The Click event notifies your application once the user presses and releases the let mouse button over the control. Unlike Click event, the <u>RClick</u> event occurs once the user right clicks the control. The Click event does not occur, if the user presses the left mouse button, drag to a new position and releases the button. By default, the control selects the event being clicked. You can use the <u>AllowSelectEvent</u> property to disable selecting the event being clicked. Use a <u>MouseDown</u> or <u>MouseUp</u> event procedure to specify actions that will occur when a mouse button is pressed or released. Unlike the Click and <u>DblClick</u> 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.

During Click event you can use the ...FromPoint(-1,-1) properties to get UI elements of the scheduler from the cursor as follows:

- Calendar. <u>DateFromPoint</u> property gets the date from the cursor in the Calendar panel.
- <u>AnchorFromPoint</u> property gets the anchor (<a>) element from the cursor.
- <u>DateTimeFromPoint</u> property gets the date/time from the cursor, in the Schedule panel.
- <u>TimeFromPoint</u> property gets the time from the cursor, in the Schedule panel.
- EventFromPoint property gets the Event object from the cursor.
- <u>GroupFromPoint</u> property retrieves the <u>Group</u> object from the cursor. Use the GroupHeaderFromPoint property to the Group object when cursor hovers the group's header.
- <u>GroupHeaderFromPoint</u> property retrieves the <u>Group</u> object from the cursor, when the cursor hovers just the group's header.
- <u>MarkTimeFromPoint</u> property gets the <u>MarkTime</u> object from the cursor. A MarkTime object is identified by a date/time where it is displaying.
- <u>MarkZoneFromPoint</u> property gets the <u>MarkZone</u> object from the cursor. A MarkZone object is identified by a starting date/time and ending date/time where it is displaying.
- <u>NonworkingTimeFromPoint</u> property indicates the <u>NonworkingTime</u> object from the cursor.
- <u>PictureFromPoint</u> property indicates the key of the picture from the cursor.
- <u>TimeScaleFromPoint</u> property returns the <u>TimeScale</u> object from the cursor.

All ... FromPoint properties that returns an Object, may return Nothing, Empty or NULL, if no object is found.

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 f	for Click event, /COM version, on:
C#	private void ClickEvent(object sender, EventArgs e) { }
C++	void OnClick() { }
C++ Builder	voidfastcall 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

\	
v	БA

Private Sub Click()	
End Sub	



Syntax f	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

The following samples display the date/time from the cursor.

VBA (MS Access, Excell...)

Click event - Occurs when the user presses and then releases the left mouse button over the control. Private Sub Schedule1_Click() With Schedule1 Debug.Print(.DateTimeFromPoint(-1,-1)) End With End Sub

VB6

' Click event - Occurs when the user presses and then releases the left mouse button over the control.

Private Sub Schedule1_Click() With Schedule1 Debug.Print(.DateTimeFromPoint(-1,-1)) End With End Sub

VB.NET

' Click event - Occurs when the user presses and then releases the left mouse button over the control.

Private Sub Exschedule1_Click(ByVal sender As System.Object) Handles Exschedule1.Click With Exschedule1 Debug.Print(.get_DateTimeFromPoint(-1,-1)) End With

End Sub

 Click event - Occurs when the user presses and then releases the left mouse button over the control.
 Private Sub AxSchedule1_ClickEvent(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles AxSchedule1.ClickEvent
 With AxSchedule1
 Debug.Print(.get_DateTimeFromPoint(-1,-1))
 End With
 End Sub

C++

```
// Click event - Occurs when the user presses and then releases the left
mouse button over the control.
void OnClickSchedule1()
{
    /*
    Copy and paste the following directives to your header file as
    it defines the namespace 'EXSCHEDULELib' for the library: 'ExSchedule 1.0
Control Library'
    #import <ExSchedule.dll>
    using namespace EXSCHEDULELib;
    */
    EXSCHEDULELib::lSchedulePtr spSchedule1 = GetDlgItem(IDC_SCHEDULE1)-
>GetControlUnknown();
    OutputDebugStringW(_bstr_t(spSchedule1->GetDateTimeFromPoint(-1,-1)) );
}
```

C++ Builder

// Click event - Occurs when the user presses and then releases the left mouse button over the control.

void __fastcall TForm1::Schedule1Click(TObject *Sender)

```
OutputDebugString( PChar(Schedule1->DateTimeFromPoint[-1,-1]) );
```

C#



JavaScript

```
<SCRIPT FOR="Schedule1" EVENT="Click()" LANGUAGE="JScript">
alert( Schedule1.DateTimeFromPoint(-1,-1) );
</SCRIPT>
<OBJECT classid="clsid:9B09E13D-7A88-4299-9DBE-383380435377"
id="Schedule1"> </OBJECT>
<SCRIPT LANGUAGE="JScript">
</SCRIPT>
```

C# for /COM

// Click event - Occurs when the user presses and then releases the left mouse button over the control.

private void axSchedule1_ClickEvent(object sender, EventArgs e)

```
System.Diagnostics.Debug.Print(
axSchedule1.get_DateTimeFromPoint(-1,-1).ToString() );
}
//this.axSchedule1.ClickEvent += new
```

EventHandler(this.axSchedule1_ClickEvent);

X++ (Dynamics Ax 2009)

```
// Click event - Occurs when the user presses and then releases the left
mouse button over the control.
void onEvent_Click()
{
    ;
    print( exschedule1.DateTimeFromPoint(-1,-1) );
}
public void init()
{
    ;
    super();
```

Delphi 8 (.NET only)

// Click event - Occurs when the user presses and then releases the left mouse button over the control.

```
procedure TWinForm1.AxSchedule1_ClickEvent(sender: System.Object; e:
System.EventArgs);
begin
  with AxSchedule1 do
  begin
    OutputDebugString( get_DateTimeFromPoint(-1,-1) );
  end
end;
```

Delphi (standard)

// Click event - Occurs when the user presses and then releases the left mouse button over the control.

procedure TForm1.Schedule1Click(ASender: TObject;); begin with Schedule1 do begin OutputDebugString(DateTimeFromPoint[-1,-1]); end end;

VFP

*** Click event - Occurs when the user presses and then releases the left mouse button over the control. *** LPARAMETERS nop with thisform.Schedule1 DEBUGOUT(.DateTimeFromPoint(-1,-1)) endwith

dBASE Plus

```
/*
with (this.ACTIVEX1.nativeObject)
   Click = class::nativeObject_Click
endwith
*/
// Occurs when the user presses and then releases the left mouse button over
the control.
function nativeObject_Click()
```

```
local oSchedule
oSchedule = form.Activex1.nativeObject
? Str(oSchedule.DateTimeFromPoint(-1,-1))
return
local oSchedule
```

```
oSchedule = form.Activex1.nativeObject
```

XBasic (Alpha Five)

```
' Occurs when the user presses and then releases the left mouse button over
the control.
function Click as v ()
Dim oSchedule as P
oSchedule = topparent:CONTROL_ACTIVEX1.activex
? oSchedule.DateTimeFromPoint(-1,-1)
end function
Dim oSchedule as P
oSchedule = topparent:CONTROL_ACTIVEX1.activex
```

Visual Objects

METHOD OCX_Exontrol1Click() CLASS MainDialog // Click event - Occurs when the user presses and then releases the left mouse button over the control.

```
OutputDebugString(String2Psz(AsString(oDCOCX_Exontrol1:
[DateTimeFromPoint,-1,-1])))
RETURN NIL
```

```
/*begin event Click() - Occurs when the user presses and then releases the left mouse
button over the control.*/
/*
    OleObject oSchedule
    oSchedule = ole_1.Object
    MessageBox("Information",string(String(oSchedule.DateTimeFromPoint(-1,-1))))
*/
/*end event Click*/
OleObject oSchedule
oSchedule = ole_1.Object
```

event DbIClick (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.

Туре	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 occurs once the user double clicks the control. By default, the control toggles the schedule view or edit the event from the cursor, when the user double click the schedule panel. In other words, if a double click occurs in the schedule view, the previously view is being displayed or inline editing the event is started. The <u>AllowToggleSchedule</u> property on exDisallow to prevent toggling the schedule view when double clicking the control. The <u>AllowEditEvent</u> property indicates the keys combination to perform the inline edit event.

During Click event you can use the ... FromPoint(-1,-1) properties to get UI elements of the scheduler from the cursor as follows:

- Calendar. <u>DateFromPoint</u> property gets the date from the cursor in the Calendar panel.
- AnchorFromPoint property gets the anchor (<a>) element from the cursor.
- <u>DateTimeFromPoint</u> property gets the date/time from the cursor, in the Schedule panel.
- <u>TimeFromPoint</u> property gets the time from the cursor, in the Schedule panel.
- EventFromPoint property gets the Event object from the cursor.
- <u>GroupFromPoint</u> property retrieves the <u>Group</u> object from the cursor. Use the GroupHeaderFromPoint property to the Group object when cursor hovers the group's header.
- <u>GroupHeaderFromPoint</u> property retrieves the <u>Group</u> object from the cursor, when the cursor hovers just the group's header.
- <u>MarkTimeFromPoint</u> property gets the <u>MarkTime</u> object from the cursor. A MarkTime object is identified by a date/time where it is displaying.
- <u>MarkZoneFromPoint</u> property gets the <u>MarkZone</u> object from the cursor. A MarkZone object is identified by a starting date/time and ending date/time where it is displaying.
- <u>NonworkingTimeFromPoint</u> property indicates the <u>NonworkingTime</u> object from the cursor.
- <u>PictureFromPoint</u> property indicates the key of the picture from the cursor.

• <u>TimeScaleFromPoint</u> property returns the <u>TimeScale</u> object from the cursor.

All ... FromPoint properties that returns an Object, may return Nothing, Empty or NULL, if no object is found.

Syntax for DblClick event, /NET version, on:

C#	private void DblClick(object sender,short Shift,int X,int Y)
	{
	}

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, AxEXSCHEDULELibIScheduleEvents_DblClickEvent e) { }
C++	void OnDblClick(short Shift,long X,long Y) { }
C++ Builder	voidfastcall 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: AxEXSCHEDULELibIScheduleEvents_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 AxEXSCHEDULELibIScheduleEvents_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(oSchedule,Shift,X,Y) RETURN
Syntax f	or 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 IIShift OLE_XPOS_PIXELS IIX OLE_YPOS_PIXELS IIY Forward Send OnComDblClick IIShift IIX IIY

End_Procedure

Visual Objects

X++

{

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

void onEvent_DblClick(int _Shift,int _X,int _Y)

XBasic	function DblClick as v (Shift as N,X as OLE::Exontrol.Schedule.1::OLE_XPOS_PIXELS,Y as OLE::Exontrol.Schedule.1::OLE_YPOS_PIXELS) end function	
dBASE	function nativeObject_DblClick(Shift,X,Y) return	

The following sample displays the date being double clicked.

VBA (MS Access, Excell...)

}

```
' DblClick event - Occurs when the user dblclk the left mouse button over an
object.
Private Sub Schedule1_DblClick(ByVal Shift As Integer,ByVal X As Long,ByVal Y As
Long)
With Schedule1
Debug.Print(.DateTimeFromPoint(-1,-1))
End With
End Sub
With Schedule1
.AllowToggleSchedule = 0
End With
```

VB6

' DblClick event - Occurs when the user dblclk the left mouse button over an object.

```
Private Sub Schedule1_DblClick(Shift As Integer,X As Single,Y As Single)
With Schedule1
Debug.Print( .DateTimeFromPoint(-1,-1) )
End With
End Sub
```

With Schedule1

```
.AllowToggleSchedule = exDisallow
End With
```

VB.NET

' DblClick event - Occurs when the user dblclk the left mouse button over an object.

Private Sub Exschedule1_DblClick(ByVal sender As System.Object,ByVal Shift As Short,ByVal X As Integer,ByVal Y As Integer) Handles Exschedule1.DblClick With Exschedule1 Debug.Print(.get_DateTimeFromPoint(-1,-1)) End With End Sub

With Exschedule1

```
.AllowToggleSchedule = exontrol.EXSCHEDULELib.AllowKeysEnum.exDisallow
End With
```

VB.NET for /COM

' DblClick event - Occurs when the user dblclk the left mouse button over an object.

Private Sub AxSchedule1_DblClick(ByVal sender As System.Object, ByVal e As AxEXSCHEDULELib._IScheduleEvents_DblClickEvent) Handles AxSchedule1.DblClick With AxSchedule1

Debug.Print(.get_DateTimeFromPoint(-1,-1))

End With

End Sub

With AxSchedule1 .AllowToggleSchedule = EXSCHEDULELib.AllowKeysEnum.exDisallow End With

C++

// DblClick event - Occurs when the user dblclk the left mouse button over an object.

void OnDblClickSchedule1(short Shift,long X,long Y)

```
{
    /*
    Copy and paste the following directives to your header file as
    it defines the namespace 'EXSCHEDULELib' for the library: 'ExSchedule 1.0
Control Library'
    #import <ExSchedule.dll>
    using namespace EXSCHEDULELib;
    */
    EXSCHEDULELib::ISchedulePtr spSchedule1 = GetDlgItem(IDC_SCHEDULE1)-
>GetControlUnknown();
    OutputDebugStringW(_bstr_t(spSchedule1->GetDateTimeFromPoint(-1,-1)));
}
EXSCHEDULELib::ISchedulePtr spSchedule1 = GetDlgItem(IDC_SCHEDULE1)-
>GetControlUnknown();
```

```
spSchedule1->PutAllowToggleSchedule(EXSCHEDULELib::exDisallow);
```

C++ Builder

// DblClick event - Occurs when the user dblclk the left mouse button over an object.

void __fastcall TForm1::Schedule1DblClick(TObject *Sender,short Shift,int X,int Y)

```
OutputDebugString( PChar(Schedule1->DateTimeFromPoint[-1,-1]) );
```

}

{

Schedule1->AllowToggleSchedule = Exschedulelib_tlb::AllowKeysEnum::exDisallow;

C#

// DblClick event - Occurs when the user dblclk the left mouse button over an object.

private void exschedule1_DblClick(object sender, short Shift, int X, int Y)

```
{
```

```
System.Diagnostics.Debug.Print(
exschedule1.get_DateTimeFromPoint(-1,-1).ToString() );
```

```
}
//this.exschedule1.DblClick += new
exontrol.EXSCHEDULELib.exg2antt.DblClickEventHandler(this.exschedule1_DblC
```

```
exschedule1.AllowToggleSchedule =
exontrol.EXSCHEDULELib.AllowKeysEnum.exDisallow;
```

JavaScript

```
<SCRIPT FOR="Schedule1" EVENT="DblClick(Shift,X,Y)" LANGUAGE="JScript">
alert(Schedule1.DateTimeFromPoint(-1,-1));
</SCRIPT>
<OBJECT classid="clsid:9B09E13D-7A88-4299-9DBE-383380435377"
id="Schedule1"></OBJECT>
<SCRIPT LANGUAGE="JScript">
Schedule1"></OBJECT>
</SCRIPT LANGUAGE="JScript">
Schedule1.AllowToggleSchedule = 0;
</SCRIPT>
```

C# for /COM

```
// DblClick event - Occurs when the user dblclk the left mouse button over an
object.
private void axSchedule1_DblClick(object sender,
AxEXSCHEDULELib_IScheduleEvents_DblClickEvent e)
{
    System.Diagnostics.Debug.Print(
    axSchedule1.get_DateTimeFromPoint(-1,-1).ToString() );
}
//this.axSchedule1.DblClick += new
AxEXSCHEDULELib_IScheduleEvents_DblClickEventHandler(this.axSchedule1_Dtext)
```

axSchedule1.AllowToggleSchedule = EXSCHEDULELib.AllowKeysEnum.exDisallow;

X++ (Dynamics Ax 2009)

```
// DblClick event - Occurs when the user dblclk the left mouse button over an
object.
void onEvent_DblClick(int _Shift,int _X,int _Y)
{
    ;
    print( exschedule1.DateTimeFromPoint(-1,-1) );
}
public void init()
{
    ;
    super();
    exschedule1.AllowToggleSchedule(0/*exDisallow*/);
}
```

Delphi 8 (.NET only)

```
// DblClick event - Occurs when the user dblclk the left mouse button over an
object.
procedure TWinForm1.AxSchedule1_DblClick(sender: System.Object; e:
AxEXSCHEDULELib_IScheduleEvents_DblClickEvent);
begin
with AxSchedule1 do
begin
OutputDebugString(get_DateTimeFromPoint(-1,-1));
end
end;
with AxSchedule1 do
begin
AllowToggleSchedule := EXSCHEDULELib.AllowKeysEnum.exDisallow;
end
```

Delphi (standard)

```
// DblClick event - Occurs when the user dblclk the left mouse button over an
object.
procedure TForm1.Schedule1DblClick(ASender: TObject; Shift : Smallint;X : Integer;Y :
Integer);
begin
with Schedule1 do
begin
OutputDebugString( DateTimeFromPoint[-1,-1] );
end
end;
with Schedule1 do
begin
AllowToggleSchedule := EXSCHEDULELib_TLB.exDisallow;
end
```

VFP

*** DblClick event - Occurs when the user dblclk the left mouse button over an object. ***

LPARAMETERS Shift,X,Y with thisform.Schedule1 DEBUGOUT(.DateTimeFromPoint(-1,-1)) endwith

with thisform.Schedule1 .AllowToggleSchedule = 0 endwith

dBASE Plus

```
/*
with (this.ACTIVEX1.nativeObject)
   DblClick = class::nativeObject_DblClick
endwith
*/
// Occurs when the user dblclk the left mouse button over an object.
function nativeObject_DblClick(Shift,X,Y)
```

```
local oSchedule
oSchedule = form.Activex1.nativeObject
? Str(oSchedule.DateTimeFromPoint(-1,-1))
return
local oSchedule
```

```
oSchedule = form.Activex1.nativeObject
oSchedule.AllowToggleSchedule = 0
```

XBasic (Alpha Five)

```
'Occurs when the user dblclk the left mouse button over an object.
function DblClick as v (Shift as N,X as OLE::Exontrol.Schedule.1::OLE_XPOS_PIXELS,Y as
OLE::Exontrol.Schedule.1::OLE_YPOS_PIXELS)
Dim oSchedule as P
oSchedule = topparent:CONTROL_ACTIVEX1.activex
? oSchedule.DateTimeFromPoint(-1,-1)
end function
Dim oSchedule as P
```

```
oSchedule = topparent:CONTROL_ACTIVEX1.activex
oSchedule.AllowToggleSchedule = 0
```

Visual Objects

```
METHOD OCX_Exontrol1DblClick(Shift,X,Y) CLASS MainDialog
```

// DblClick event - Occurs when the user dblclk the left mouse button over an object.

```
OutputDebugString(String2Psz(AsString(oDCOCX_Exontrol1:
```

```
[DateTimeFromPoint,-1,-1]) ))
```

RETURN NIL

oDCOCX_Exontrol1:AllowToggleSchedule := exDisallow

PowerBuilder

```
/*begin event DblClick(integer Shift,long X,long Y) - Occurs when the user dblclk the
left mouse button over an object.*/
/*
    OleObject oSchedule
    oSchedule = ole_1.Object
    MessageBox("Information",string( String(oSchedule.DateTimeFromPoint(-1,-1)) ))
*/
/*end event DblClick*/
OleObject oSchedule
    oSchedule = ole_1.Object
    oSchedule = ole_1.Object
    oSchedule.AllowToggleSchedule = 0
```

event Error (Error as Long, Description as String)

Fired when an internal error occurs.

Туре	Description
Error as Long	A long expression that indicates the error number.
Description as String	A string expression that describes the error.

The Error event occurs when an internal error occurs. The Error event is usually fired when the control is bounded to an ADO/DAO Recordset. For instance, if the user changes a field, the control tries to update the current record. If it fails, the Error event is fired. The <u>DataSource</u> property binds the control to a database.

Syntax for Error event, /NET version, on:

C#	private void Error(object sender, int Err, string Description)	
	{	
	}	
\/D	Private Sub Error(ByVal sender As System Object ByVal Err As Integer ByVal	

Private Sub Error(ByVal sender As System.Object,ByVal Err As Integer,ByVal Description As String) Handles Error End Sub

Syntax for Error event, /COM version, on:

C#	private void Error(object sender, AxEXSCHEDULELibIScheduleEvents_ErrorEvent e) { }
C++	void OnError(long Error,LPCTSTR Description) { }
C++ Builder	voidfastcall Error(TObject *Sender,long Error,BSTR Description) { }
Delphi	procedure Error(ASender: TObject; Error : Integer;Description : WideString); begin

	end;
Delphi 8 (.NET only)	procedure Error(sender: System.Object; e: AxEXSCHEDULELibIScheduleEvents_ErrorEvent); begin end;
Powe	begin event Error(long Error,string Description) end event Error
VB.NET	Private Sub Error(ByVal sender As System.Object, ByVal e As AxEXSCHEDULELibIScheduleEvents_ErrorEvent) Handles Error End Sub
VB6	Private Sub Error(ByVal Error As Long,ByVal Description As String) End Sub
VBA	Private Sub Error(ByVal Error As Long,ByVal Description As String) End Sub
VFP	LPARAMETERS Error, Description
Xbas	PROCEDURE OnError(oSchedule,Error,Description) RETURN
Syntax f	for Error event, /COM version ^(others) , on:
Java	<script event="Error(Error,Description)" language="JScript"> </script>
VBSc	<script language="VBScript"> Function Error(Error,Description) End Function </script>
Visual Data	Procedure OnComError Integer IIError String IIDescription

Forward Send OnComError IIError IIDescription

End_Procedure

Visual Objects	METHOD OCX_Error(Error,Description) CLASS MainDialog RETURN NIL
X++	void onEvent_Error(int _Error,str _Description) { }
XBasic	function Error as v (Error as N,Description as C) end function
dBASE	function nativeObject_Error(Error,Description) return

event Event (EventID as Long)

Notifies the application once the control fires an event.

Туре	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 <u>EventParam</u> (-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. For instance if a RClick event occurs, then a Event(1) occurs also, so the events inside the Event are differentiated by its EventID. Print the EventParam(-2) during the Event notification, and you get debugging information for the name, ID, and parameters of the fired event. For instance, <u>Click</u> event has no parameter, which means that the EventParam(-1) gets 0.

Click here \blacksquare to watch a movie on how you can use the <u>eXHelper</u> to get information about the fired events using the Event handler. The Event notification is sent any time the control fires a specified event. For instance, if the BarResize event occurs, the order of the events are Event(120) and next BarResize. You can use any of these notifications based on your requirements or limitations of the programming environment you are using.

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 exschedule1.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,=6,=false)" VT_BSTR

"AfterDrawPart/55(2,-1962866148,0,0,0,0)" VT_BSTR

"MouseMove/-606(1,0,145,35)" VT_BSTR
```

Each line indicates an event, and the following information is provided: the name of the event, its identifier, and the list of parameters being passed to the event. The parameters that starts with = character, indicates a parameter by reference, in other words one that can changed during the event handler.

Now, we can see that the identifier for the BarParentChange event is 125, so we need to handle the Event event as:

```
// Notifies the application once the control fires an event.
void onEvent_Event(int _EventID)
{
    ;
    if ( _EventID == 125 ) /*event BarParentChange (Item as HITEM, Key as Variant, NewItem
as HITEM, Cancel as Boolean) */
    exschedule1.EventParam( 3 /*Cancel*/, COMVariant::createFromBoolean(true) );
}
```

The code checks if the BarParentChange (_EventID == 125) event is fired, and changes the third parameter of the event to true. The definition for BarParentChange event can be consulted in the control's documentation or in the ActiveX explorer. So, anytime you need to access the original parameters for the event you should use the EventParam method that allows you to get or set a parameter. If the parameter is not passed by reference, you can not change the parameter's value.

Now, let's add some code to see a complex sample, so let's say that we need to prevent moving the bar from an item to any disabled item. So, we need to specify the Cancel

parameter as not Items.EnableItem(NewItem), in other words cancels if the new parent is disabled. Shortly the code will be:

```
// Notifies the application once the control fires an event.
void onEvent_Event(int _EventID)
{
    ;
    if ( _EventID == 125 ) /*event BarParentChange (Item as HITEM, Key as Variant, NewItem
as HITEM, Cancel as Boolean) */
    if ( !exschedule1.Items().EnableItem( exschedule1.EventParam( 2 /*NewItem*/ ) ) )
        exschedule1.EventParam( 3 /*Cancel*/, COMVariant::createFromBoolean(true) );
}
```

In conclusion, anytime the X++ fires the "invalid parameters." while handling an event, you can use and handle the Event notification and EventParam methods of the control

Syntax for Event event, /NET version, on:

C#	private void Event(object sender, int EventID)
	{
	}
VB	Private Sub Event(ByVal sender As System.Object,ByVal EventID As Integer)

Handles Event

End Sub

Syntax for Event event, /COM version, on:

C#	private void Event(object sender, AxEXSCHEDULELibIScheduleEvents_EventEvent e) { }
C++	void OnEvent(long EventID) { }
C++ Builder	voidfastcall Event(TObject *Sender,long EventID) { }

Delphi	procedure Event(ASender: TObject; EventID : Integer); begin end;
Delphi 8 (.NET only)	procedure Event(sender: System.Object; e: AxEXSCHEDULELibIScheduleEvents_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 AxEXSCHEDULELibIScheduleEvents_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(oSchedule,EventID) RETURN
Syntax f	for Event event, / COM version ^(others) , on:
Java	<script event="Event(EventID)" language="JScript"> </script>
VBSc	<script language="VBScript"></script>

End Function </SCRIPT>



Procedure OnComEvent Integer IIEventID Forward Send OnComEvent IIEventID End_Procedure	
Visual Objects	METHOD OCX_Event(EventID) CLASS MainDialog RETURN NIL
X++	void onEvent_Event(int _EventID) { }
XBasic	function Event as v (EventID as N) end function
dBASE	function nativeObject_Event(EventID) return

event KeyDown (ByRef KeyCode as Integer, Shift as Integer)

Occurs when the user presses a key while an object has the focus.

Туре	Description
KeyCode as Integer	(By Reference) An integer that represent the key code.
Shift as Integer	An integer that corresponds to the state of the SHIFT, CTRL, and ALT keys at the time of the event. The shift argument is a bit field with the least-significant bits corresponding to the SHIFT key (bit 0), the CTRL key (bit 1), and the ALT key (bit 2). These bits correspond to the values 1, 2, and 4, respectively. Some, all, or none of the bits can be set, indicating that some, all, or none of the keys are pressed. For example, if both CTRL and ALT are pressed, the value of shift is 6.

Use KeyDown and KeyUp event procedures if you need to respond to both the pressing and releasing of a key. You test for a condition by first assigning each result to a temporary integer variable and then comparing shift to a bit mask. Use the And operator with the shift argument to test whether the condition is greater than 0, indicating that the modifier was pressed, as in this example:

ShiftDown = (Shift And 1) > 0 CtrlDown = (Shift And 2) > 0 AltDown = (Shift And 4) > 0 In a procedure, you can test for any combination of conditions, as in this example: If AltDown And CtrlDown Then

Syntax for KeyDown event, /NET version, on:



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:

)#	private void KeyDownEvent(object sender,
	AxEXSCHEDULELibIScheduleEvents_KeyDownEvent e

	{ }
C++	void OnKeyDown(short FAR* KeyCode,short Shift) { }
C++ Builder	voidfastcall 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: AxEXSCHEDULELibIScheduleEvents_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 AxEXSCHEDULELibIScheduleEvents_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(oSchedule,KeyCode,Shift) RETURN

L
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 IIKeyCode Short IIShift Forward Send OnComKeyDown IIKeyCode IIShift End_Procedure
Visual Objects	METHOD OCX_KeyDown(KeyCode,Shift) CLASS MainDialog RETURN NIL
X++	<pre>void onEvent_KeyDown(COMVariant /*short*/ _KeyCode,int _Shift) { }</pre>
XBasic	function KeyDown as v (KeyCode as N,Shift as N) end function
dBASE	function nativeObject_KeyDown(KeyCode,Shift) return

event KeyPress (ByRef KeyAscii as Integer)

Occurs when the user presses and releases an ANSI key.

Туре	Description
KeyAscii as Integer	(By Reference) An integer that returns a standard numeric ANSI keycode.

The KeyPress event lets you immediately test keystrokes for validity or for formatting characters as they are typed. Changing the value of the keyascii argument changes the character displayed. Use <u>KeyDown</u> and <u>KeyUp</u> 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

Syntax for KeyPress event, /COM version, on:

End Sub

Builde

C#	private void KeyPressEvent(object sender, AxEXSCHEDULELibIScheduleEvents_KeyPressEvent e) { }
C++	void OnKeyPress(short FAR* KeyAscii) { }

void __fastcall KeyPress(TObject *Sender, short * KeyAscii) C++

Delphi	procedure KeyPress(ASender: TObject; var KeyAscii : Smallint); begin end;
Dolphi Q	procedure KeyPressEvent(sender: System Object: e
(.NET	AxEXSCHEDULELib IScheduleEvents KeyPressEvent)
only)	begin
	end:
Powe	begin event KeyPress(integer KeyAscii)
	end event KeyPress
VB.NET	Private Sub KeyPressEvent(ByVal sender As System.Object, ByVal e As
	AxEXSCHEDULELibIScheduleEvents_KeyPressEvent) Handles KeyPressEvent
	End Sub
VB6	Private Sub KeyPress(KeyAscii As Integer)
	End Sub
	Private Sub KeyPress(KeyAscii As Integer)
VBA	Fnd Sub
VFP	LPARAMETERS KeyAscii
لكلنفي	
Xbas	PROCEDURE OnKeyPress(oSchedule,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 IIKeyAscii Forward Send OnComKeyPress IIKeyAscii End_Procedure
Visual Objects	METHOD OCX_KeyPress(KeyAscii) CLASS MainDialog RETURN NIL
X++	void onEvent_KeyPress(COMVariant /*short*/ _KeyAscii) { }
XBasic	function KeyPress as v (KeyAscii as N) end function
dBASE	function nativeObject_KeyPress(KeyAscii) return

event KeyUp (ByRef KeyCode as Integer, Shift as Integer)

Occurs when the user releases a key while an object has the focus.

Туре	Description
KeyCode as Integer	(By Reference) An integer that represent the key code.
Shift as Integer	An integer that corresponds to the state of the SHIFT, CTRL, and ALT keys at the time of the event. The shift argument is a bit field with the least-significant bits corresponding to the SHIFT key (bit 0), the CTRL key (bit 1), and the ALT key (bit 2). These bits correspond to the values 1, 2, and 4, respectively. Some, all, or none of the bits can be set, indicating that some, all, or none of the keys are pressed. For example, if both CTRL and ALT are pressed, the value of shift is 6.

Use the KeyUp event procedure to respond to the releasing of a key.

Syntax for KeyUp event, /NET version, on:

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

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:

Builder

C#	private void KeyUpEvent(object sender, AxEXSCHEDULELibIScheduleEvents_KeyUpEvent e) { }
C++	void OnKeyUp(short FAR* KeyCode,short Shift) { }

Delphi procedure KeyUp(ASender: TObject; var KeyCode : Smallint;Shift : Smallint); begin end; Delphi 8 procedure KeyUpEvent(sender: System.Object; e: AxEXSCHEDULELibIScheduleEvents_KeyUpEvent); begin end; Powe begin event KeyUp(integer KeyCode,integer Shift) end event KeyUp Powe Private Sub KeyUpEvent(ByVal sender As System.Object, ByVal e As AxEXSCHEDULELibIScheduleEvents_KeyUpEvent) Handles KeyUpEvent End Sub
Delphi 8 (NET only) procedure KeyUpEvent(sender: System.Object; e: AxEXSCHEDULELibIScheduleEvents_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 AxEXSCHEDULELibIScheduleEvents_KeyUpEvent) Handles KeyUpEvent End Sub
Powe begin event KeyUp(integer KeyCode,integer Shift) end event KeyUp VB.NET Private Sub KeyUpEvent(ByVal sender As System.Object, ByVal e As AxEXSCHEDULELibIScheduleEvents_KeyUpEvent) Handles KeyUpEvent End Sub
VB.NET Private Sub KeyUpEvent(ByVal sender As System.Object, ByVal e As AxEXSCHEDULELibIScheduleEvents_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(oSchedule,KeyCode,Shift) RETURN

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

Java...

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

/BSc	<script language="VBScript"></script>
------	---------------------------------------

	End Function
Visual Data	Procedure OnComKeyUp Short IIKeyCode Short IIShift Forward Send OnComKeyUp IIKeyCode IIShift 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 LayoutEndChanging (Operation as LayoutChangingEnum)

Notifies your application once the control's layout has been changed.

Туре	Description
Operation as	A LayoutChangingEnum expression that indicates the
<u>LayoutChangingEnum</u>	operation that ends .

Generally, the LayoutEndChanging event occurs once an UI operation is performed (ended). For instance, if the user resizes an event the LayoutEndChanging(exScheduleResizeStartEvent) or LayoutEndChanging(exScheduleResizeEndEvent) may occurs once the user resized an event. Once the operation starts the LayoutStartChanging with the same code (operation) is fired. Another sample of using the LayoutStartChanging/LayoutEndChanging events, is to be notified once the user changes the selected dates in the calendar panel or in the schedule view, using the exCalendarSelectionChange or exScheduleSelectionChange. Generally speaking, a LayoutEndChanging(Operation) is preceded by a LayoutStartChanging(Operation).

The LayoutEndChanging event may occurs in the following situations:

- *exLayoutResizePanels(0)*, one of the panels is resized. The <u>PaneWidth</u> property indicates the width of the left/right panel.
- exCalendarSelectionChange(1), specifies whether the selection in the calendar panel is changed. The <u>Selection</u> property of the Calendar returns a safe array of selected dates. The /NET or /WPF version provides the SelDates property of List<DateTime> type to get or sets the new selection using a collection of DateTime objects. The Calendar.<u>SelCount</u> and Calendar.<u>SelDate</u> properties gives the selection dates. The LayoutEndChanging(exCalendarSelectionChange) notifies once the selection of dates in the calendar section is changed.
- exCalendarFocusDateChange(2), specifies whether the focused date in the calendar panel is changed. The <u>FocusDate</u> property indicates the date being focused in the calendar.
- *exCalendarDateChange(3)*, notifies whether the browsing date in the calendar panel is changed. The <u>Date</u> property indicates the month date being browsed in the calendar.
- exScheduleMove(4), notifies once the user moved the schedule view to a new position by drag and drop. By default, you can press the SHIFT + Click and drag the schedule view to a new position. The <u>AllowMoveSchedule</u> property indicates the keys combination so the user can move the schedule to a new position.
- *exScheduleResize(5)*, occurs once the schedule view is resized. By default, you can click the middle mouse button, and drag the cursor to a new position, so the schedule view gets zoomed or resized. The <u>AllowResizeSchedule</u> property indicates the keys combination so the user can resize the schedule view at runtime.
- exScheduleResizeTimeScale(6), notifies once the user is resized the control's time

scale. The <u>TimeScales</u> property access the control's <u>TimeScale</u> objects.

- exLayoutCalendarAutoHide(7), notifies your application once the calendar panel is shown or hidden. The OnResizeControl property on exCalendarAutoHide makes the calendar goes away if the cursor is not in it. The PaneWidth property indicates the width of the left/right panel. For instance, the PaneWidth(False) on 0, indicates that the calendar panel is hidden, or if it not zero, the calendar panel is shown. You can call the FitSelToView method during this operation so the schedule fits the selected dates in its client area.
- exScheduleCreateEvent(8), occurs once a new event is created using the mouse. The AddEvent event notifies your application once a new event is added to the schedule view.
- exScheduleResizeGroup(9), specifies whether the user resized a group. The user can resize a group by clicking the groups header between two groups, and start dragging the cursor to a new position, and so the group is being resized.
- exScheduleSelectionChange(10), indicates whether the user is selected events in the schedule panel. The <u>Selection</u> property gets or sets a safe array of selected events. The /NET or /WPF version provides the SelEvents property of List<Event> type to get or sets the new selection using a collection of Event objects. The Selected property of the Event indicates whether the current event is selected or unselected. The Selectable property indicates whether a specified event can be selected at runtime. The AllowSelectEvent property indicates the combination of the keys to let user selects the events.
- exScheduleMoveEvent(11), indicates whether the user is about to move events in the schedule panel. The control fires the UpdateEvent event once the margin of the events are being updated.
- exScheduleResizeStartEvent(12), indicates whether the user resized the starting point of the event. The control fires the UpdateEvent event once the margin of the events are being updated.
- exScheduleResizeEndEvent(13), indicates whether the user resized the ending point of the event. The control fires the UpdateEvent event once the margin of the events are being updated.
- exScheduleMoveMarkTime(14), indicates whether the user moved a MarkTime object. The Movable property of the MarkTime object indicates whether the user can move at runtime the MarkTime object using the Mouse. The MarkTimeFromPoint(-1,-1) property indicates the MarkTime object from the cursor.

Use the <u>Parent</u> property to move the calendar panel to other place.

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

private void LayoutEndChanging(object C# sender, exontrol. EXSCHEDULELib. Layout Changing Enum Operation)

	}
VB	Private Sub LayoutEndChanging(ByVal sender As System.Object,ByVal Operation As exontrol.EXSCHEDULELib.LayoutChangingEnum) Handles LayoutEndChanging End Sub
Syntax f	for LayoutEndChanging event, / COM version, on:
C#	private void LayoutEndChanging(object sender, AxEXSCHEDULELibIScheduleEvents_LayoutEndChangingEvent e) { }
C++	void OnLayoutEndChanging(long Operation) { }
C++ Builder	<pre>voidfastcall LayoutEndChanging(TObject *Sender,Exschedulelib_tlb::LayoutChangingEnum Operation) { }</pre>
Delphi	procedure LayoutEndChanging(ASender: TObject; Operation : LayoutChangingEnum); begin end;
Delphi 8 (.NET only)	procedure LayoutEndChanging(sender: System.Object; e: AxEXSCHEDULELibIScheduleEvents_LayoutEndChangingEvent); begin end;
Powe	begin event LayoutEndChanging(long Operation) end event LayoutEndChanging
VB.NET	Private Sub LayoutEndChanging(ByVal sender As System.Object, ByVal e As AxEXSCHEDULELibIScheduleEvents_LayoutEndChangingEvent) Handles LayoutEndChanging

	End Sub
VB6	Private Sub LayoutEndChanging(ByVal Operation As EXSCHEDULELibCtl.LayoutChangingEnum) End Sub
VBA	Private Sub LayoutEndChanging(ByVal Operation As Long) End Sub
VFP	LPARAMETERS Operation
Xbas	PROCEDURE OnLayoutEndChanging(oSchedule,Operation) RETURN
Syntax f	or LayoutEndChanging event, / COM version ^(others) , on:
Java	<script event="LayoutEndChanging(Operation)" language="JScript"> </script>
VBSc	<script language="VBScript"> Function LayoutEndChanging(Operation) End Function </script>
Visual Data	Procedure OnComLayoutEndChanging OLELayoutChangingEnum IIOperation Forward Send OnComLayoutEndChanging IIOperation End_Procedure
Visual Objects	METHOD OCX_LayoutEndChanging(Operation) CLASS MainDialog RETURN NIL
X++	void onEvent_LayoutEndChanging(int _Operation) { }
XBasic	function LayoutEndChanging as v (Operation as OLE::Exontrol.Schedule.1::LayoutChangingEnum)

end function

dBASE function nativeObject_LayoutEndChanging(Operation) return

The following two samples shows how to use a separate calendar control, instead the schedule's calendar panel:

• Using the <u>eXCalendar</u> as a separate control

Insert the CalendarCombo or eXCalendar to a form, and name it Calendar1. Next, add the eXSchedule component to the same form, and name it to Schedule1. Now, use the following code:

```
Private Sub Calendar1_SelectionChanged()
Schedule1.Calendar.Selection = Calendar1.Value
End Sub
Private Sub Form_Load()
With Schedule1
.OnResizeControl = 768
.Calendar.Selection = Calendar1.Value
End With
End Sub
```

If you want so synchronize the dates, so once the user clicks another date in the schedule view, you can use a code as follows:

```
Private Sub Schedule1_MouseDown(Button, Shift, X, Y)

If (Shift = 0) Then

If (Button = 1) Then

With Schedule1

Dim d As Date

d = Fix(.DateTimeFromPoint(-1, -1))

If (d <> 0) Then

Calendar1.Value = d

End If

End With

End If
```

```
End If
End Sub
```

SHIFT + Click the schedule view, and move the view to a new position, and then click. The new date in the calendar is selected. The *Fix(.DateTimeFromPoint(-1, -1))* gets the integer part of the date, in other words, just the date from the cursor, not including the time section. The *.DateTimeFromPoint(-1, -1)* gets the date including the time being clicked.

• Using the Calendar Panel of the ExSchedule component as a separate control.

Insert the eXSchedule control to a form, and name it Calendar1. Next, add a new eXSchedule to the same form, and name it to Schedule1. Now, use the following code:

```
Private Sub Calendar1_LayoutEndChanging(Operation)

If (Operation = exCalendarSelectionChange) Then

Schedule1.Calendar.Selection = Calendar1.Calendar.Selection

End If

End Sub

Private Sub Form_Load()

With Calendar1

.OnResizeControl = 257

End With

With Schedule1

.OnResizeControl = 768

.Calendar.Selection = Calendar1.Calendar.Selection

End With

End Sub
```

If you want so synchronize the dates, so once the user clicks another date in the schedule view, you can use a code as follows:

```
Private Sub Schedule1_MouseDown(Button, Shift, X, Y)

If (Shift = 0) Then

If (Button = 1) Then

With Schedule1

Dim d As Date

d = Fix(.DateTimeFromPoint(-1, -1))
```

```
If (d <> 0) Then
Calendar1.Calendar.Selection = d
End If
End With
End If
End If
End If
End Sub
```

SHIFT + Click the schedule view, and move the view to a new position, and then click. The new date in the calendar is selected. The *Fix(.DateTimeFromPoint(-1, -1))* gets the integer part of the date, in other words, just the date from the cursor, not including the time section. The *.DateTimeFromPoint(-1, -1)* gets the date including the time being clicked.

The samples changes the <u>Selection</u> property of the Schedule's Calendar object, when the selection is changed in the second (the calendar) control.

event LayoutStartChanging (Operation as LayoutChangingEnum)

Occurs when the control's layout is about to be changed.

Туре	Description
Operation as	A LayoutChangingEnum expression that indicates the
LayoutChangingEnum	operation is about to begin.

Generally, the LayoutStartChanging event occurs once an UI operation is performed (started). For instance, if the user resizes an event the

LayoutStartChanging(exScheduleResizeStartEvent) or

LayoutStartChanging(exScheduleResizeEndEvent) may occurs once the user starts resizing an event. Once the operation ends the <u>LayoutEndChanging</u> with the same code (operation) is fired. Another sample of using the LayoutStartChanging/LayoutEndChanging events, is to be notified once the user changes the selected dates in the calendar panel or in the schedule view, using the exCalendarSelectionChange or exScheduleSelectionChange. Generally speaking, a LayoutStartChanging(Operation) is followed by a LayoutEndChanging(Operation).

The LayoutStartChanging event may occurs in the following situations:

- *exLayoutResizePanels(0)*, one of the panels is being resized. The <u>PaneWidth</u> property indicates the width of the left/right panel.
- exCalendarSelectionChange(1), specifies whether the selection in the calendar panel is changing. The <u>Selection</u> property of the Calendar returns a safe array of selected dates. The /NET or /WPF version provides the SelDates property of List<DateTime> type to get or sets the new selection using a collection of DateTime objects.
- exCalendarFocusDateChange(2), specifies whether the focused date in the calendar panel is changing. The FocusDate property indicates the date being focused in the calendar. The AllowFocusDate property specifies the combination of keys that allows the user to focus a new date, in the calendar panel. The Background(exCalendarFocusDate) changes the visual appearance of the focused date, while the Background(exCalendarFocusDateForeColor) changes the foreground color of the focused date.
- exCalendarDateChange(3), notifies whether the browsing date in the calendar panel is changing. The <u>Date</u> property indicates the month date being browsed in the calendar. The <u>FirstVisibleDate/LastVisibleDate</u> property indicates the first visible date in the calendar panel.
- exScheduleMove(4), notifies once the user is about to move the schedule view to a new position by drag and drop. By default, you can press the SHIFT + Click and drag the schedule view to a new position. The <u>AllowMoveSchedule</u> property indicates the keys combination so the user can move the schedule to a new position.
- *exScheduleResize*(5), occurs once the schedule view is resizing. By default, you can

click the middle mouse button, and drag the cursor to a new position, so the schedule view gets zoomed or resized. The <u>AllowResizeSchedule</u> property indicates the keys combination so the user can resize the schedule view at runtime.

- exScheduleResizeTimeScale(6), notifies once the user is resizing the control's time scale. The <u>TimeScales</u> property access the control's <u>TimeScale</u> objects.
- exLayoutCalendarAutoHide(7), notifies your application once the calendar panel is shown or hidden. The <u>OnResizeControl</u> property on exCalendarAutoHide makes the calendar goes away if the cursor is not in it. The <u>PaneWidth</u> property indicates the width of the left/right panel. For instance, the PaneWidth(False) on 0, indicates that the calendar panel is hidden, or if it not zero, the calendar panel is shown. You can call the <u>FitSelToView</u> method during this operation so the schedule fits the selected dates in its client area.
- exScheduleCreateEvent(8), occurs once a new event is creating using the mouse. The <u>AddEvent</u> event notifies your application once a new event is added to the schedule view.
- *exScheduleResizeGroup(9)*, specifies whether the user is resizing a group. The user can resize a group by clicking the groups header between two groups, and start dragging the cursor to a new position, and so the group is being resized.
- exScheduleSelectionChange(10), indicates whether the user is about to select events in the schedule panel. The <u>Selection</u> property gets or sets a safe array of selected events. The /NET or /WPF version provides the SelEvents property of List<Event> type to get or sets the new selection using a collection of Event objects.
- exScheduleMoveEvent(11), indicates whether the user is about to move events in the schedule panel. The control fires the <u>UpdateEvent</u> event once the margin of the events are being updated.
- exScheduleResizeStartEvent(12), indicates whether the user is about to resize the starting point of the event. The control fires the <u>UpdateEvent</u> event once the margin of the events are being updated.
- exScheduleResizeEndEvent(13), indicates whether the user is about to resize the ending point of the event. The control fires the <u>UpdateEvent</u> event once the margin of the events are being updated.
- exScheduleMoveMarkTime(14), indicates whether the user is about to move a <u>MarkTime</u> object. The Movable property of the MarkTime object indicates whether the user can move at runtime the MarkTime object using the Mouse. The <u>MarkTimeFromPoint</u>(-1,-1) property indicates the <u>MarkTime</u> object from the cursor.
- exScheduleEditEvent(15), indicates whether the user edits the event's caption. The event notifies once the user starts inline editing an appointment. The Editable property of the Event indicates the property of the Event to be edited at runtime. You can use the EventFromPoint(-1,-1) method during the LayoutStartChanging(exScheduleEditEvent) to store the event from the cursor to a global member, and when LayoutEndChanging(exScheduleEditEvent) occurs, you can use the previously stored member to identify the event being edited.

• exScheduleMoveGroup,(18),

VB

Syntax for LayoutStartChanging event, /NET version, on:

C# private void LayoutStartChanging(object sender,exontrol.EXSCHEDULELib.LayoutChangingEnum Operation) { }

Private Sub LayoutStartChanging(ByVal sender As System.Object,ByVal Operation As exontrol.EXSCHEDULELib.LayoutChangingEnum) Handles LayoutStartChanging End Sub

Syntax for LayoutStartChanging event, /COM version, on:

C#	private void LayoutStartChanging(object sender,
	AxEXSCHEDULELibIScheduleEvents_LayoutStartChangingEvent e)
	{
	}
C++	void OnLayoutStartChanging(long Operation)
	{
	}
C++	voidfastcall LayoutStartChanging(TObject
Builder	*Sender,Exschedulelib_tlb::LayoutChangingEnum Operation)
	{
	}
Delphi	procedure LayoutStartChanging(ASender: TObject; Operation :
	LayoutChangingEnum);
	begin
	end;
Delphi 8	procedure LayoutStartChanging(sender: System.Object; e:
(.NET	AxEXSCHEDULELib_IScheduleEvents_LavoutStartChangingEvent)·

begin

only)

end;

Powe	
begin end ev	event LayoutStartChanging(long Operation) /ent LayoutStartChanging
VB.NET	Private Sub LayoutStartChanging(ByVal sender As System.Object, ByVal e As AxEXSCHEDULELibIScheduleEvents_LayoutStartChangingEvent) Handles LayoutStartChanging End Sub
VB6	Private Sub LayoutStartChanging(ByVal Operation As EXSCHEDULELibCtl.LayoutChangingEnum) End Sub
VBA	Private Sub LayoutStartChanging(ByVal Operation As Long) End Sub
VFP	LPARAMETERS Operation
Xbas	PROCEDURE OnLayoutStartChanging(oSchedule,Operation) RETURN
Syntax	for LayoutStartChanging event, / COM version ^(others) , on:
Java	<script event="LayoutStartChanging(Operation)" language="JScript"> </script>
VBSc	<script language="VBScript"> Function LayoutStartChanging(Operation) End Function </script>
Visual Data…	Procedure OnComLayoutStartChanging OLELayoutChangingEnum IIOperation Forward Send OnComLayoutStartChanging IIOperation End_Procedure
Visual	METHOD OCX_LayoutStartChanging(Operation) CLASS MainDialog

Objects RETURN NIL

X++	void onEvent_LayoutStartChanging(int _Operation) { }
XBasic	function LayoutStartChanging as v (Operation as OLE::Exontrol.Schedule.1::LayoutChangingEnum) end function
dBASE	function nativeObject_LayoutStartChanging(Operation) 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.

Туре	Description
Button as Integer	An integer that identifies the button that was pressed to cause the event as 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 when the button specified in the button argument is pressed or released.
X as OLE_XPOS_PIXELS	A single that specifies the current X location of the mouse pointer. The X value is always expressed in container coordinates.
Y as OLE_YPOS_PIXELS	A single that specifies the current Y location of the mouse pointer. The Y value is always expressed in container coordinates.

Use a MouseDown or <u>MouseUp</u> event procedure to specify actions that will occur when a mouse button is pressed or released. Unlike the <u>Click</u> and <u>DblClick</u> 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.

During Click event you can use the ... FromPoint(-1,-1) properties to get UI elements of the scheduler from the cursor as follows:

- Calendar. <u>DateFromPoint</u> property gets the date from the cursor in the Calendar panel.
- AnchorFromPoint property gets the anchor (<a>) element from the cursor.
- <u>DateTimeFromPoint</u> property gets the date/time from the cursor, in the Schedule panel.
- <u>TimeFromPoint</u> property gets the time from the cursor, in the Schedule panel.
- EventFromPoint property gets the Event object from the cursor.
- <u>GroupFromPoint</u> property retrieves the <u>Group</u> object from the cursor. Use the GroupHeaderFromPoint property to the Group object when cursor hovers the group's header.
- <u>GroupHeaderFromPoint</u> property retrieves the <u>Group</u> object from the cursor, when the cursor hovers just the group's header.
- <u>MarkTimeFromPoint</u> property gets the <u>MarkTime</u> object from the cursor. A MarkTime object is identified by a date/time where it is displaying.
- <u>MarkZoneFromPoint</u> property gets the <u>MarkZone</u> object from the cursor. A MarkZone object is identified by a starting date/time and ending date/time where it is displaying.

- <u>NonworkingTimeFromPoint</u> property indicates the <u>NonworkingTime</u> object from the cursor.
- <u>PictureFromPoint</u> property indicates the key of the picture from the cursor.
- <u>TimeScaleFromPoint</u> property returns the <u>TimeScale</u> object from the cursor.

All ... FromPoint properties that returns an Object, may return Nothing, Empty or NULL, if no object is found.

Use the <u>Parent</u> property to move the calendar panel to other place.

Syntax for MouseDown event, /NET version, on:

VB

{ }

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

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,
	AxEXSCHEDULELibIScheduleEvents_MouseDownEvent e)
	{
	}

c++ void OnMouseDown(short Button,short Shift,long X,long Y)

	void fasteall Mayza Dawn (TObject *Condershort Dutten short Chift int Vint V)
C++	voldlastcall wouseDown(TObject "Sender, short button, short Shirt, int X, int Y)
Builder	{
	}
	,

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: AxEXSCHEDULELibIScheduleEvents_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 AxEXSCHEDULELibIScheduleEvents_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(oSchedule,Button,Shift,X,Y) RETURN
Syntax f	or 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 IIButton Short IIShift OLE_XPOS_PIXELS IIX OLE_YPOS_PIXELS IIY

Forward Send OnComMouseDown IIButton IIShift IIX IIY 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.Schedule.1::OLE_XPOS_PIXELS,Y as OLE::Exontrol.Schedule.1::OLE_YPOS_PIXELS) end function
dBASE	function nativeObject_MouseDown(Button,Shift,X,Y) return

The following two samples shows how to use a separate calendar control, instead the schedule's calendar panel:

• Using the <u>eXCalendar</u> as a separate control

Insert the CalendarCombo or eXCalendar to a form, and name it Calendar1. Next, add the eXSchedule component to the same form, and name it to Schedule1. Now, use the following code:

```
Private Sub Calendar1_SelectionChanged()
Schedule1.Calendar.Selection = Calendar1.Value
End Sub
Private Sub Form_Load()
With Schedule1
.OnResizeControl = 768
.Calendar.Selection = Calendar1.Value
End With
End Sub
```

If you want so synchronize the dates, so once the user clicks another date in the

schedule view, you can use a code as follows:

```
Private Sub Schedule1_MouseDown(Button, Shift, X, Y)

If (Shift = 0) Then

If (Button = 1) Then

With Schedule1

Dim d As Date

d = Fix(.DateTimeFromPoint(-1, -1))

If (d <> 0) Then

Calendar1.Value = d

End If

End With

End If

End If

End Sub
```

SHIFT + Click the schedule view, and move the view to a new position, and then click. The new date in the calendar is selected. The *Fix(.DateTimeFromPoint(-1, -1))* gets the integer part of the date, in other words, just the date from the cursor, not including the time section. The *.DateTimeFromPoint(-1, -1)* gets the date including the time being clicked.

• Using the Calendar Panel of the ExSchedule component as a separate control.

Insert the eXSchedule control to a form, and name it Calendar1. Next, add a new eXSchedule to the same form, and name it to Schedule1. Now, use the following code:

```
Private Sub Calendar1_LayoutEndChanging(Operation)
If (Operation = exCalendarSelectionChange) Then
Schedule1.Calendar.Selection = Calendar1.Calendar.Selection
End If
End Sub
Private Sub Form_Load()
With Calendar1
.OnResizeControl = 257
End With
With Schedule1
```

```
.OnResizeControl = 768
.Calendar.Selection = Calendar1.Calendar.Selection
End With
End Sub
```

If you want so synchronize the dates, so once the user clicks another date in the schedule view, you can use a code as follows:

```
Private Sub Schedule1_MouseDown(Button, Shift, X, Y)

If (Shift = 0) Then

If (Button = 1) Then

With Schedule1

Dim d As Date

d = Fix(.DateTimeFromPoint(-1, -1))

If (d <> 0) Then

Calendar1.Calendar.Selection = d

End If

End With

End If

End If

End If

End Sub
```

SHIFT + Click the schedule view, and move the view to a new position, and then click. The new date in the calendar is selected. The *Fix(.DateTimeFromPoint(-1, -1))* gets the integer part of the date, in other words, just the date from the cursor, not including the time section. The *.DateTimeFromPoint(-1, -1)* gets the date including the time being clicked.

The samples changes the <u>Selection</u> property of the Schedule's Calendar object, when the selection is changed in the second (the calendar) control.

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.

Туре	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.

The MouseMove event is generated continually as the mouse pointer moves across objects. Unless another object has captured the mouse, an object recognizes a MouseMove event whenever the mouse position is within its borders. During the MouseMove event you can call the <u>ShowToolTip</u> method to display any custom tooltip. For instance, you can display a tooltip when the cursor is hovering an anchor element <a> like shown in the bellow samples:

During MouseMove event you can use the ...FromPoint(-1,-1) properties to get UI elements of the scheduler from the cursor as follows:

- Calendar. <u>DateFromPoint</u> property gets the date from the cursor in the Calendar panel.
- AnchorFromPoint property gets the anchor (<a>) element from the cursor.
- <u>DateTimeFromPoint</u> property gets the date/time from the cursor, in the Schedule panel.
- <u>TimeFromPoint</u> property gets the time from the cursor, in the Schedule panel.
- EventFromPoint property gets the Event object from the cursor.
- <u>GroupFromPoint</u> property retrieves the <u>Group</u> object from the cursor. Use the GroupHeaderFromPoint property to the Group object when cursor hovers the group's header.
- <u>GroupHeaderFromPoint</u> property retrieves the <u>Group</u> object from the cursor, when the cursor hovers just the group's header.
- <u>MarkTimeFromPoint</u> property gets the <u>MarkTime</u> object from the cursor. A MarkTime object is identified by a date/time where it is displaying.
- <u>MarkZoneFromPoint</u> property gets the <u>MarkZone</u> object from the cursor. A MarkZone object is identified by a starting date/time and ending date/time where it is displaying.

- <u>NonworkingTimeFromPoint</u> property indicates the <u>NonworkingTime</u> object from the cursor.
- <u>PictureFromPoint</u> property indicates the key of the picture from the cursor.
- <u>TimeScaleFromPoint</u> property returns the <u>TimeScale</u> object from the cursor.

All ... FromPoint properties that returns an Object, may return Nothing, Empty or NULL, if no object is found.

Syntax for MouseMove event, /NET version, on:

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

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,
	AxEXSCHEDULELibIScheduleEvents_MouseMoveEvent e)
	{
	}

C++	void OnMouseMove(short Button,short Shift,long X,long Y)
	{
	}

C++ Builder	voidfastcall 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;



proceo AxEXS begin end;	dure MouseMoveEvent(sender: System.Object; e: CHEDULELibIScheduleEvents_MouseMoveEvent);
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 AxEXSCHEDULELibIScheduleEvents_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(oSchedule,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" th=""></script>	
---	--

</SCRIPT>

Visual Data	Procedure OnComMouseMove Short IIButton Short IIShift OLE_XPOS_PIXELS IIX OLE_YPOS_PIXELS IIY Forward Send OnComMouseMove IIButton IIShift IIX IIY End_Procedure
Visual Objects	METHOD OCX_MouseMove(Button,Shift,X,Y) CLASS MainDialog RETURN NIL
X++	<pre>Void OnEvent_IVIOUSelVIOVe(Int _Button,Int _Shift,Int _X,Int _Y) { }</pre>
	function MousoMovo as V (Button as N Shift as N V as
XBasic	OI E"Exontrol Schedule 1"OI E XPOS PIXELS V as
	OLEExontrol.Schedule 1::OLE_XI OS_LIXELS, 1 as
	end function
dBASE	function nativeObject_MouseMove(Button,Shift,X,Y) return

The following samples show how to show a custom tooltip when the cursor hovers to an anchor <a> element.

VBA (MS Access, Excell...)

```
' MouseMove event - Occurs when the user moves the mouse.
Private Sub Schedule1_MouseMove(ByVal Button As Integer,ByVal Shift As
Integer,ByVal X As Long,ByVal Y As Long)
With Schedule1
    s = .AnchorFromPoint(-1,-1)
    .ShowToolTip s,"info",""," + 16"
End With
End Sub
```

```
' MouseMove event - Occurs when the user moves the mouse.
Private Sub Schedule1_MouseMove(Button As Integer,Shift As Integer,X As Single,Y
As Single)
With Schedule1
s = .AnchorFromPoint(-1,-1)
.ShowToolTip s,"info","","+16"
End With
End Sub
```

VB.NET

```
' MouseMove event - Occurs when the user moves the mouse.
Private Sub Exschedule1_MouseMoveEvent(ByVal sender As System.Object,ByVal
Button As Short,ByVal Shift As Short,ByVal X As Integer,ByVal Y As Integer) Handles
Exschedule1.MouseMoveEvent
Dim s
With Exschedule1
    s = .get_AnchorFromPoint(-1,-1)
    .ShowToolTip(s,"info","","+16")
End With
End Sub
```

VB.NET for /COM

```
' MouseMove event - Occurs when the user moves the mouse.
Private Sub AxSchedule1_MouseMoveEvent(ByVal sender As System.Object, ByVal e
As AxEXSCHEDULELib._IScheduleEvents_MouseMoveEvent) Handles
AxSchedule1.MouseMoveEvent
Dim s
With AxSchedule1
s = .get_AnchorFromPoint(-1,-1)
.ShowToolTip(s,"info","","+16")
End With
End Sub
```

C++

```
// MouseMove event - Occurs when the user moves the mouse.
void OnMouseMoveSchedule1(short Button,short Shift,long X,long Y)
{
    /*
        Copy and paste the following directives to your header file as
        it defines the namespace 'EXSCHEDULELib' for the library: 'ExSchedule 1.0
Control Library'
        #import <ExSchedule.dll>
        using namespace EXSCHEDULELib;
        */
        EXSCHEDULELib::ISchedulePtr spSchedule1 = GetDlgItem(IDC_SCHEDULE1)-
>GetControlUnknown();
        _bstr_t s = spSchedule1->GetAnchorFromPoint(-1,-1);
        spSchedule1->ShowToolTip(L"s","info","","+16",vtMissing);
    }
```

C++ Builder

```
// MouseMove event - Occurs when the user moves the mouse.
void __fastcall TForm1::Schedule1MouseMove(TObject *Sender,short Button,short
Shift,int X,int Y)
{
    String s = Schedule1->AnchorFromPoint[-1,-1];
    Schedule1-
>ShowToolTip(L"s",TVariant("info"),TVariant(""),TVariant("+16"),TNoParam());
}
```

```
// MouseMove event - Occurs when the user moves the mouse.
private void exschedule1_MouseMoveEvent(object sender,short Button,short Shift,int
X,int Y)
{
    string s = exschedule1.get_AnchorFromPoint(-1,-1);
    exschedule1.ShowToolTip(s.ToString(),"info","","+16",null);
}
//this.exschedule1.MouseMoveEvent += new
exontrol.EXSCHEDULELib.exg2antt.MouseMoveEventHandler(this.exschedule1_levented)
```

JavaScript

```
<SCRIPT FOR="Schedule1" EVENT="MouseMove(Button,Shift,X,Y)"
LANGUAGE="JScript">
var s = Schedule1.AnchorFromPoint(-1,-1);
Schedule1.ShowToolTip(s,"info","","+16",null);
</SCRIPT>
<OBJECT classid="clsid:9B09E13D-7A88-4299-9DBE-383380435377"
id="Schedule1"></OBJECT>
```

```
<SCRIPT LANGUAGE="JScript">
</SCRIPT>
```

C# for /COM

```
// MouseMove event - Occurs when the user moves the mouse.
private void axSchedule1_MouseMoveEvent(object sender,
AxEXSCHEDULELib_IScheduleEvents_MouseMoveEvent e)
{
    string s = axSchedule1.get_AnchorFromPoint(-1,-1);
    axSchedule1.ShowToolTip(s.ToString(),"info","","+16",null);
}
//this.axSchedule1.MouseMoveEvent += new
AxEXSCHEDULELib_IScheduleEvents_MouseMoveEventHandler(this.axSchedule
```

X++ (Dynamics Ax 2009)

```
// MouseMove event - Occurs when the user moves the mouse.
void onEvent_MouseMove(int _Button,int _Shift,int _X,int _Y)
{
    str s;
    ;
    s = exschedule1.AnchorFromPoint(-1,-1);
    exschedule1.ShowToolTip(s,"info","","+16");
}
public void init()
{
    ;
    super();
}
```

Delphi 8 (.NET only)

```
// MouseMove event - Occurs when the user moves the mouse.
procedure TWinForm1.AxSchedule1_MouseMoveEvent(sender: System.Object; e:
AxEXSCHEDULELib._IScheduleEvents_MouseMoveEvent);
begin
  with AxSchedule1 do
  begin
    s := get_AnchorFromPoint(-1,-1);
    ShowToolTip(s,'info','','+16',Nil);
  end
end;
```

Delphi (standard)

```
// MouseMove event - Occurs when the user moves the mouse.
procedure TForm1.Schedule1MouseMove(ASender: TObject; Button : Smallint;Shift :
Smallint;X : Integer;Y : Integer);
begin
    with Schedule1 do
    begin
    s := AnchorFromPoint[-1,-1];
    ShowToolTip(s,'info','','+16',Null);
    end
end;
```

VFP

*** MouseMove event - Occurs when the user moves the mouse. ***
LPARAMETERS Button,Shift,X,Y
with thisform.Schedule1
s = .AnchorFromPoint(-1,-1)
.ShowToolTip(s,"info","","+16")
endwith

dBASE Plus

```
/*
with (this.ACTIVEX1.nativeObject)
   MouseMove = class::nativeObject_MouseMove
endwith
*/
// Occurs when the user moves the mouse.
function nativeObject_MouseMove(Button,Shift,X,Y)
   local oSchedule,s
   oSchedule = form.Activex1.nativeObject
   s = oSchedule.AnchorFromPoint(-1,-1)
```

```
oSchedule.ShowToolTip(Str(s),"info","","+16")
return
```

local oSchedule

```
oSchedule = form.Activex1.nativeObject
```

XBasic (Alpha Five)

```
'Occurs when the user moves the mouse.
function MouseMove as v (Button as N,Shift as N,X as
OLE::Exontrol.Schedule.1::OLE_XPOS_PIXELS,Y as
OLE::Exontrol.Schedule.1::OLE_YPOS_PIXELS)
Dim oSchedule as P
Dim s as
oSchedule = topparent:CONTROL_ACTIVEX1.activex
s = oSchedule.AnchorFromPoint(-1,-1)
oSchedule.ShowToolTip(s,"info",""," + 16")
end function
Dim oSchedule as P
```

oSchedule = topparent:CONTROL_ACTIVEX1.activex

Visual Objects

METHOD OCX_Exontrol1MouseMove(Button,Shift,X,Y) CLASS MainDialog // MouseMove event - Occurs when the user moves the mouse. local s as USUAL s := oDCOCX_Exontrol1:[AnchorFromPoint,-1,-1] oDCOCX_Exontrol1:ShowToolTip(AsString(s),"info","","+16",nil) RETURN NIL

PowerBuilder

```
/*begin event MouseMove(integer Button,integer Shift,long X,long Y) - Occurs when
the user moves the mouse.*/
/*
    OleObject oSchedule
    any s
    oSchedule = ole_1.Object
    s = oSchedule.AnchorFromPoint(-1,-1)
    oSchedule.ShowToolTip(String(s),"info","","+16")
*/
/*end event MouseMove*/
OleObject oSchedule
oSchedule = ole_1.Object
```
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.

Туре	Description
Button as Integer	An integer that identifies the button that was pressed to cause the event as 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 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 <u>MouseDown</u> or MouseUp event procedure to specify actions that will occur when a mouse button is pressed or released. Unlike the <u>Click</u> and <u>DblClick</u> 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.

During Click event you can use the ... FromPoint(-1,-1) properties to get UI elements of the scheduler from the cursor as follows:

- Calendar. <u>DateFromPoint</u> property gets the date from the cursor in the Calendar panel.
- AnchorFromPoint property gets the anchor (<a>) element from the cursor.
- <u>DateTimeFromPoint</u> property gets the date/time from the cursor, in the Schedule panel.
- <u>TimeFromPoint</u> property gets the time from the cursor, in the Schedule panel.
- EventFromPoint property gets the Event object from the cursor.
- <u>GroupFromPoint</u> property retrieves the <u>Group</u> object from the cursor. Use the GroupHeaderFromPoint property to the Group object when cursor hovers the group's header.
- <u>GroupHeaderFromPoint</u> property retrieves the <u>Group</u> object from the cursor, when the cursor hovers just the group's header.
- <u>MarkTimeFromPoint</u> property gets the <u>MarkTime</u> object from the cursor. A MarkTime object is identified by a date/time where it is displaying.
- <u>MarkZoneFromPoint</u> property gets the <u>MarkZone</u> object from the cursor. A MarkZone object is identified by a starting date/time and ending date/time where it is displaying.

- <u>NonworkingTimeFromPoint</u> property indicates the <u>NonworkingTime</u> object from the cursor.
- <u>PictureFromPoint</u> property indicates the key of the picture from the cursor.
- <u>TimeScaleFromPoint</u> property returns the <u>TimeScale</u> object from the cursor.

All ... FromPoint properties that returns an Object, may return Nothing, Empty or NULL, if no object is found.

Syntax for MouseUp event, /NET version, on:

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

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,
	AxEXSCHEDULELibIScheduleEvents_MouseUpEvent e)
	{
	}

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

C++ Builder	voidfastcall 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;



proce AxEXS begin end;	dure MouseUpEvent(sender: System.Object; e: CHEDULELibIScheduleEvents_MouseUpEvent);
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 AxEXSCHEDULELibIScheduleEvents_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(oSchedule,Button,Shift,X,Y) RETURN

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

Java	<script event="MouseUp(Button,Shift,X,Y)" language="JScript"></th></tr><tr><th></th><th></script>
------	---

VBSc	<pre>sc <script language="VBScript"></pre></th></tr><tr><th></th><th>Function MouseUp(Button,Shift,X,Y)</th></tr><tr><th></th><th>End Function</th></tr><tr><th></th><th></script></pre>	
------	--	--



Procedure OnComMouseUp Short IIButton Short IIShift OLE_XPOS_PIXELS IIX
OLE_YPOS_PIXELS IIY

Forward Send OnComMouseUp IIButton IIShift IIX IIY

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.Schedule.1::OLE_XPOS_PIXELS,Y as OLE::Exontrol.Schedule.1::OLE_YPOS_PIXELS) end function
dBASE	function nativeObject_MouseUp(Button,Shift,X,Y) return

event OLECompleteDrag (Effect as Long)

Occurs when a source component is dropped onto a target component, informing the source component that a drag action was either performed or canceled

Туре	Description
Effect as Long	A long set by the source object identifying the action that has been performed, thus allowing the source to take appropriate action if the component was moved (such as the source deleting data if it is moved from one component to another

The OLECompleteDrag event is the final event to be called in an OLE drag/drop operation. This event informs the source component of the action that was performed when the object was dropped onto the target component. The target sets this value through the effect parameter of the <u>OLEDragDrop</u> event. Based on this, the source can then determine the appropriate action it needs to take. For example, if the object was moved into the target (exDropEffectMove), the source needs to delete the object from itself after the move. The control supports only manual OLE drag and drop events. In order to enable OLE drag and drop feature into control you have to set the <u>OLEDropMode</u> and <u>OLEDrag</u> properties.

The settings for Effect are:

- exOLEDropEffectNone (0), Drop target cannot accept the data, or the drop operation was cancelled
- exOLEDropEffectCopy (1), Drop results in a copy of data from the source to the target. The original data is unaltered by the drag operation.
- exOLEDropEffectMove (2), Drop results in data being moved from drag source to drop source. The drag source should remove the data from itself after the move.

Syntax for OLECompleteDrag event, /NET version, on:

// OLECompleteDrag event is not supported. Use the DragEnter,DragLeave,DragOver, DragDrop ... events.

VB

C#

C#

// OLECompleteDrag event is not supported. Use the DragEnter,DragLeave,DragOver, DragDrop ... events.

Syntax for OLECompleteDrag event, /COM version, on:

private void OLECompleteDrag(object sender, AxEXSCHEDULELib._IScheduleEvents_OLECompleteDragEvent e)

	}
C++	void OnOLECompleteDrag(long Effect) { }
C++ Builder	voidfastcall OLECompleteDrag(TObject *Sender,long Effect) { }
Delphi	procedure OLECompleteDrag(ASender: TObject; Effect : Integer); begin end;
Delphi 8 (.NET only)	procedure OLECompleteDrag(sender: System.Object; e: AxEXSCHEDULELibIScheduleEvents_OLECompleteDragEvent); begin end;
Powe	begin event OLECompleteDrag(long Effect) end event OLECompleteDrag
VB.NET	Private Sub OLECompleteDrag(ByVal sender As System.Object, ByVal e As AxEXSCHEDULELibIScheduleEvents_OLECompleteDragEvent) Handles OLECompleteDrag End Sub
VB6	Private Sub OLECompleteDrag(ByVal Effect As Long) End Sub
VBA	Private Sub OLECompleteDrag(ByVal Effect As Long) End Sub
VFP	LPARAMETERS Effect
Xbas	PROCEDURE OnOLECompleteDrag(oSchedule,Effect) RETURN

Syntax f	or OLECompleteDrag event, / COM version ^(others) , on:
Java	<script event="OLECompleteDrag(Effect)" language="JScript"> </script>
VBSc	<script language="VBScript"> Function OLECompleteDrag(Effect) End Function </script>
Visual Data	Procedure OnComOLECompleteDrag Integer IIEffect Forward Send OnComOLECompleteDrag IIEffect End_Procedure
Visual Objects	METHOD OCX_OLECompleteDrag(Effect) CLASS MainDialog RETURN NIL
X++	// OLECompleteDrag event is not supported. Use the DragEnter,DragLeave,DragOver, DragDrop events.
XBasic	function OLECompleteDrag as v (Effect as N) end function
dBASE	function nativeObject_OLECompleteDrag(Effect) return

event OLEDragDrop (Data as ExDataObject, Effect as Long, Button as Integer, Shift as Integer, X as OLE_XPOS_PIXELS, Y as OLE_YPOS_PIXELS)

Occurs when a source component is dropped onto a target component when the source component determines that a drop can occur.

Туре	Description
Data as <u>ExDataObject</u>	An ExDataObject object containing formats that the source will provide and, in addition, possibly the data for those formats. If no data is contained in the ExDataObject, it is provided when the control calls the GetData method. The SetData and Clear methods cannot be used here.
Effect as Long	A Long set by the target component identifying the action that has been performed (if any), thus allowing the source to take appropriate action if the component was moved (such as the source deleting the data). The possible values are listed in Remarks.
Button as Integer	An integer which acts as a bit field corresponding to the state of a mouse button when it is depressed. The left button is bit 0, the right button is bit 1, and the middle button is bit 2. These bits correspond to the values 1, 2, and 4, respectively. It indicates the state of the mouse buttons; some, all, or none of these three bits can be set, indicating that some, all, or none of the buttons are depressed.
Shift as Integer	An integer which acts as a bit field corresponding to the state of the SHIFT, CTRL, and ALT keys when they are depressed. The SHIFT key is bit 0, the CTRL key is bit 1, and the ALT key is bit 2. These bits correspond to the values 1, 2, and 4, respectively. The shift parameter indicates the state of these keys; some, all, or none of the bits can be set, indicating that some, all, or none of the keys are depressed. For example, if both the CTRL and ALT keys were depressed, the value of shift would be 6.
X as OLE_XPOS_PIXELS	A single that specifies the current X location of the mouse pointer. The X value is always expressed in container coordinates
Y as OLE_YPOS_PIXELS	A single that specifies the current Y location of the mouse pointer. The Y value is always expressed in container coordinates.

In the /NET Assembly, you have to use the DragDrop event as explained here:

• <u>https://www.exontrol.com/sg.jsp?content=support/faq/net/#dragdrop</u>

The OLEDragDrop event is fired when the user has dropped files or clipboard information into the control. Use the <u>OLEDropMode</u> property on exOLEDropManual to enable OLE drop and drop support. Use the <u>EventFromPoint</u> property to get the event from point. Use the <u>DateTimeFromPoint</u> property to get the date-time from point in the schedule part of the control. Use the <u>DateFromPoint</u> property to retrieve the date from the cursor over the calendar section of the control. Use the <u>Add</u> method to add a event to the control. The <u>Background(exScheduleOLEDropPosition)</u> property specifies the visual appearance of the line to be shown when the cursor is hovering the schedule part of the control, during an OLE drag and drop operation.

The settings for Effect are:

C#

- exOLEDropEffectNone (0), Drop target cannot accept the data, or the drop operation was cancelled
- exOLEDropEffectCopy (1), Drop results in a copy of data from the source to the target. The original data is unaltered by the drag operation.
- exOLEDropEffectMove (2), Drop results in data being moved from drag source to drop source. The drag source should remove the data from itself after the move.

Syntax for OLEDragDrop event, /NET version, on:

- // OLEDragDrop event is not supported. Use the DragEnter,DragLeave,DragOver, DragDrop ... events.
- VB // OLEDragDrop event is not supported. Use the DragEnter,DragLeave,DragOver, DragDrop ... events.

Syntax for OLEDragDrop event, /COM version, on:

private void OLEDragDrop(object sender, AxEXSCHEDULELib._IScheduleEvents_OLEDragDropEvent e)

C++ void OnOLEDragDrop(LPDISPATCH Data,long FAR* Effect,short Button,short Shift,long X,long Y)

C++ Builder	<pre>voidfastcall OLEDragDrop(TObject *Sender,Exschedulelib_tlb::IExDataObject *Data,long * Effect,short Button,short Shift,int X,int Y) { }</pre>
Delphi	procedure OLEDragDrop(ASender: TObject; Data : IExDataObject;var Effect : Integer;Button : Smallint;Shift : Smallint;X : Integer;Y : Integer); begin end;
Delphi 8 (.NET only)	procedure OLEDragDrop(sender: System.Object; e: AxEXSCHEDULELibIScheduleEvents_OLEDragDropEvent); begin end;
Powe	begin event OLEDragDrop(oleobject Data,long Effect,integer Button,integer Shift,long X,long Y) end event OLEDragDrop
VB.NET	Private Sub OLEDragDrop(ByVal sender As System.Object, ByVal e As AxEXSCHEDULELibIScheduleEvents_OLEDragDropEvent) Handles OLEDragDrop End Sub
VB6	Private Sub OLEDragDrop(ByVal Data As EXSCHEDULELibCtl.IExDataObject,Effect As Long,ByVal Button As Integer,ByVal Shift As Integer,ByVal X As Single,ByVal Y As Single) End Sub
VBA	Private Sub OLEDragDrop(ByVal Data As Object,Effect As Long,ByVal Button As Integer,ByVal Shift As Integer,ByVal X As Long,ByVal Y As Long) End Sub
VFP	LPARAMETERS Data, Effect, Button, Shift, X, Y
Xbas	PROCEDURE OnOLEDragDrop(oSchedule,Data,Effect,Button,Shift,X,Y) RETURN

```
Syntax for OLEDragDrop event, /COM version (others), on:
```

-	
Java	<script <br="" event="OLEDragDrop(Data,Effect,Button,Shift,X,Y)">LANGUAGE="JScript"> </script>
VBSc	<script language="VBScript"> Function OLEDragDrop(Data,Effect,Button,Shift,X,Y) End Function </script>
Visual Data	Procedure OnComOLEDragDrop Variant IIData Integer IIEffect Short IIButton Short IIShift OLE_XPOS_PIXELS IIX OLE_YPOS_PIXELS IIY Forward Send OnComOLEDragDrop IIData IIEffect IIButton IIShift IIX IIY End_Procedure
Visual Objects	METHOD OCX_OLEDragDrop(Data,Effect,Button,Shift,X,Y) CLASS MainDialog RETURN NIL
Х++	// OLEDragDrop event is not supported. Use the DragEnter,DragLeave,DragOver, DragDrop events.
XBasic	function OLEDragDrop as v (Data as OLE::Exontrol.Schedule.1::IExDataObject,Effect as N,Button as N,Shift as N,X as OLE::Exontrol.Schedule.1::OLE_XPOS_PIXELS,Y as OLE::Exontrol.Schedule.1::OLE_YPOS_PIXELS) end function
dBASE	function nativeObject_OLEDragDrop(Data,Effect,Button,Shift,X,Y) return

The following VB sample adds a new event for each dragged file (Open the Windows Explorer, click and drag a file to the control) :

Private Sub Schedule1_OLEDragDrop(ByVal Data As EXSCHEDULELibCtl.IExDataObject, Effect As Long, ByVal Button As Integer, ByVal Shift As Integer, ByVal X As Single, ByVal Y As Single)

If Data.GetFormat(exCFFiles) Then

```
With Schedule1

Dim d As Date

d = .DateTimeFromPoint(-1, -1)

If Not (d = 0) Then

With .Events

Dim f As Variant

For Each f In Data.Files

.Add(d, d + 2 / 24).ExtraLabel = f

Next

End With

End If

End With

End If

End Sub
```

The sample queries the date-time over the schedule part of the control, and add a new event for each dragged file.

event OLEDragOver (Data as ExDataObject, Effect as Long, Button as Integer, Shift as Integer, X as OLE_XPOS_PIXELS, Y as OLE_YPOS_PIXELS, State as Integer)

Occurs when one component is dragged over another.

Туре	Description
Data as <u>ExDataObject</u>	An ExDataObject object containing formats that the source will provide and, in addition, possibly the data for those formats. If no data is contained in the ExDataObject, it is provided when the control calls the GetData method. The SetData and Clear methods cannot be used here
Effect as Long	A Long set by the target component identifying the action that has been performed (if any), thus allowing the source to take appropriate action if the component was moved (such as the source deleting the data). The possible values are listed in Remarks.
Button as Integer	An integer which acts as a bit field corresponding to the state of a mouse button when it is depressed. The left button is bit 0, the right button is bit 1, and the middle button is bit 2. These bits correspond to the values 1, 2, and 4, respectively. It indicates the state of the mouse buttons; some, all, or none of these three bits can be set, indicating that some, all, or none of the buttons are depressed.
Shift as Integer	An integer which acts as a bit field corresponding to the state of the SHIFT, CTRL, and ALT keys when they are depressed. The SHIFT key is bit 0, the CTRL key is bit 1, and the ALT key is bit 2. These bits correspond to the values 1, 2, and 4, respectively. The shift parameter indicates the state of these keys; some, all, or none of the bits can be set, indicating that some, all, or none of the keys are depressed. For example, if both the CTRL and ALT keys were depressed, the value of shift would be 6.
X as OLE_XPOS_PIXELS	A single that specifies the current X location of the mouse pointer. The X value is always expressed in container coordinates.
Y as OLE_YPOS_PIXELS	A single that specifies the current Y location of the mouse pointer. The Y value is always expressed in container coordinates.

The <u>Background(exScheduleOLEDropPosition</u>) property specifies the visual appearance of the line to be shown when the cursor is hovering the schedule part of the control, during an OLE drag and drop operation.

The settings for effect are:

- exOLEDropEffectNone (0), Drop target cannot accept the data, or the drop operation was cancelled
- exOLEDropEffectCopy (1), Drop results in a copy of data from the source to the target. The original data is unaltered by the drag operation.
- exOLEDropEffectMove (2), Drop results in data being moved from drag source to drop source. The drag source should remove the data from itself after the move.

The settings for state are:

- exOLEDragEnter (0), Source component is being dragged within the range of a target.
- exOLEDragLeave (1), Source component is being dragged out of the range of a target.
- exOLEOLEDragOver (2), Source component has moved from one position in the target to another.

Note If the state parameter is 1, indicating that the mouse pointer has left the target, then the x and y parameters will contain zeros.

The source component should always mask values from the effect parameter to ensure compatibility with future implementations of ActiveX components. As a precaution against future problems, drag sources and drop targets should mask these values appropriately before performing any comparisons.

For example, a source component should not compare an effect against, say, exOLEDropEffectCopy, such as in this manner:

If Effect = exOLEDropEffectCopy...

Instead, the source component should mask for the value or values being sought, such as this:

If Effect And exOLEDropEffectCopy = exOLEDropEffectCopy...

-or-

If (Effect And exOLEDropEffectCopy)...

This allows for the definition of new drop effects in future versions while preserving backwards compatibility with your existing code.

The control supports only manual OLE drag and drop events.

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

Oynax i	or olebrago vor ovork, met voroion, on.
C#	<pre>// OLEDragOver event is not supported. Use the DragEnter,DragLeave,DragOver, DragDrop events.</pre>
VB	<pre>// OLEDragOver event is not supported. Use the DragEnter,DragLeave,DragOver, DragDrop events.</pre>
Syntax f	or OLEDragOver event, / COM version, on:
C#	private void OLEDragOver(object sender,
	AxEXSCHEDULELibIScheduleEvents_OLEDragOverEvent e) { }
C++	void OnOLEDragOver(LPDISPATCH Data,long FAR* Effect,short Button,short Shift,long X,long Y,short State) { }
	-
C++ Builder	voidfastcall OLEDragOver(TObject *Sender,Exschedulelib_tlb::IExDataObject *Data,long * Effect,short Button,short Shift,int X,int Y,short State) { }
Delphi	procedure OLEDragOver(ASender: TObject; Data : IExDataObject;var Effect : Integer;Button : Smallint;Shift : Smallint;X : Integer;Y : Integer;State : Smallint); begin end;
Delphi 8 (.NET only)	procedure OLEDragOver(sender: System.Object; e: AxEXSCHEDULELibIScheduleEvents_OLEDragOverEvent); begin end;
Powe	begin event OLEDragOver(oleobject Data,long Effect,integer Button,integer Shift,long X,long Y,integer State) end event OLEDragOver
· · · · · ·	5

VB.NET	
Private AxEXS End St	e Sub OLEDragOver(ByVal sender As System.Object, ByVal e As CHEDULELibIScheduleEvents_OLEDragOverEvent) Handles OLEDragOver ub
	Private Sub OLEDragOver(BvVal Data As EXSCHEDLILELibCtLEvDataObject Effect
VB6	As Long,ByVal Button As Integer,ByVal Shift As Integer,ByVal X As Single,ByVal Y As Single,ByVal State As Integer) End Sub
	Private Sub OLEDragOver(BvVal Data As Object Effect As Long BvVal Button As
VBA	Integer,ByVal Shift As Integer,ByVal X As Long,ByVal Y As Long,ByVal State As Integer) End Sub
	LDADANATTERS Data Effect Button Shift V V State
VFP	LPARAMETERS Data, Effect, Button, Shirt, X, Y, State
	DDOCEDURE On OLEDna a Over (a Calcala Data Effect Detter Chift V.) (Ctata)
Xbas	RETURN
Syntax	for OLEDragOver event, / COM version ^(others) , on:
Java	<script <="" event="OLEDragOver(Data,Effect,Button,Shift,X,Y,State)" td=""></tr><tr><td></td><td>LANGUAGE="JScript"></td></tr><tr><td></td><td></script>
VBSc	<script language="VBScript"></td></tr><tr><td></td><td>Function OLEDragOver(Data,Effect,Button,Shift,X,Y,State)</td></tr><tr><td></td><td>End Function</td></tr><tr><td></td><td></script>
	Procedure OnComOLEDragOver Variant IIData Integer IIEffect Short IIButton Short
Visual Data…	
	Forward Send OnComOLEDragOver IIData IIEffect IIButton IIShift IIX IIY IIState End_Procedure
	۱



METHOD OCX_OLEDragOver(Data,Effect,Button,Shift,X,Y,State) CLASS MainDialog
RETURN NILX++// OLEDragOver event is not supported. Use the DragEnter,DragLeave,DragOver,
DragDrop ... events.XBasicfunction OLEDragOver as v (Data as OLE::Exontrol.Schedule.1::IExDataObject,Effect
as N,Button as N,Shift as N,X as OLE::Exontrol.Schedule.1::OLE_XPOS_PIXELS,Y as
OLE::Exontrol.Schedule.1::OLE_YPOS_PIXELS,State as N)
end functiondBASEfunction nativeObject_OLEDragOver(Data,Effect,Button,Shift,X,Y,State)
return

event OLEGiveFeedback (Effect as Long, DefaultCursors as Boolean)

Allows the drag source to specify the type of OLE drag-and-drop operation and the visual feedback.

Туре	Description
Effect as Long	A long integer set by the target component in the OLEDragOver event specifying the action to be performed if the user drops the selection on it. This allows the source to take the appropriate action (such as giving visual feedback). The possible values are listed in Remarks.
DefaultCursors as Boolean	Boolean value that determines whether to use the default mouse cursor, or to use a user-defined mouse cursor.True (default) = use default mouse cursor.False = do not use default cursor. Mouse cursor must be set with the MousePointer property of the Screen object.

The <u>Background(exScheduleOLEDropPosition</u>) property specifies the visual appearance of the line to be shown when the cursor is hovering the schedule part of the control, during an OLE drag and drop operation.

The settings for Effect are:

- exOLEDropEffectNone (0), Drop target cannot accept the data, or the drop operation was cancelled
- exOLEDropEffectCopy (1), Drop results in a copy of data from the source to the target. The original data is unaltered by the drag operation.
- exOLEDropEffectMove (2), Drop results in data being moved from drag source to drop source. The drag source should remove the data from itself after the move.

If there is no code in the OLEGiveFeedback event, or if the defaultcursors parameter is set to True, the mouse cursor will be set to the default cursor provided by the control. The source component should always mask values from the effect parameter to ensure compatibility with future implementations of ActiveX components. As a precaution against future problems, drag sources and drop targets should mask these values appropriately before performing any comparisons.

For example, a source component should not compare an effect against, say,

exOLEDropEffectCopy, such as in this manner:

If Effect = exOLEDropEffectCopy...

Instead, the source component should mask for the value or values being sought, such as this:

If Effect And exOLEDropEffectCopy = exOLEDropEffectCopy...

-or-

C#

If (Effect And exOLEDropEffectCopy)...

This allows for the definition of new drop effects in future versions while preserving backwards compatibility with your existing code.

The control supports only manual OLE drag and drop events.

Syntax for OLEGiveFeedback event, /NET version, on:

// OLEGiveFeedback event is not supported. Use the DragEnter,DragLeave,DragOver, DragDrop ... events.

VB // OLEGiveFeedback event is not supported. Use the DragEnter,DragLeave,DragOver, DragDrop ... events.

Syntax for OLEGiveFeedback event, /COM version, on:

-	
C#	private void OLEGiveFeedback(object sender, AxEXSCHEDULELibIScheduleEvents_OLEGiveFeedbackEvent e) { }
C++	void OnOLEGiveFeedback(long Effect,BOOL FAR* DefaultCursors) { }
C++ Builder	voidfastcall OLEGiveFeedback(TObject *Sender,long Effect,VARIANT_BOOL * DefaultCursors) { }
Delphi	procedure OLEGiveFeedback(ASender: TObject; Effect : Integer;var DefaultCursors : WordBool); begin end;
Delphi 8 (.NET only)	procedure OLEGiveFeedback(sender: System.Object; e: AxEXSCHEDULELibIScheduleEvents_OLEGiveFeedbackEvent); begin end:

Powe	begin event OLEGiveFeedback(long Effect,boolean DefaultCursors) end event OLEGiveFeedback
VB.NET	Private Sub OLEGiveFeedback(ByVal sender As System.Object, ByVal e As AxEXSCHEDULELibIScheduleEvents_OLEGiveFeedbackEvent) Handles OLEGiveFeedback End Sub
VB6	Private Sub OLEGiveFeedback(ByVal Effect As Long,DefaultCursors As Boolean) End Sub
VBA	Private Sub OLEGiveFeedback(ByVal Effect As Long,DefaultCursors As Boolean) End Sub
VFP	LPARAMETERS Effect, Default Cursors
Xbas	PROCEDURE OnOLEGiveFeedback(oSchedule,Effect,DefaultCursors) RETURN
Syntax	for OLEGiveFeedback event, / COM version ^(others) , on:
Java	<script <br="" event="OLEGiveFeedback(Effect,DefaultCursors)">LANGUAGE="JScript"> </script>
VBSc	<script language="VBScript"> Function OLEGiveFeedback(Effect,DefaultCursors) End Function </script>
Visual Data	Procedure OnComOLEGiveFeedback Integer IIEffect Boolean IIDefaultCursors Forward Send OnComOLEGiveFeedback IIEffect IIDefaultCursors End_Procedure
Visual Objects	METHOD OCX_OLEGiveFeedback(Effect,DefaultCursors) CLASS MainDialog RETURN NIL

X++

// OLEGiveFeedback event is not supported. Use the DragEnter,DragLeave,DragOver, DragDrop ... events.

XBasic	function OLEGiveFeedback as v (Effect as N,DefaultCursors as L)
	end function

dBASE

function nativeObject_OLEGiveFeedback(Effect,DefaultCursors) return

event OLESetData (Data as ExDataObject, Format as Integer)

Occurs on a drag source when a drop target calls the GetData method and there is no data in a specified format in the OLE drag-and-drop DataObject.

Туре	Description
Data as <u>ExDataObject</u>	An ExDataObject object in which to place the requested data. The component calls the SetData method to load the requested format.
Format as Integer	An integer specifying the format of the data that the target component is requesting. The source component uses this value to determine what to load into the ExDataObject object.

The OLESetData is not currently supported.

Syntax for OLESetData event, /NET version, on:

C# // OLESetData event is not supported. Use the DragEnter,DragLeave,DragOver, DragDrop ... events.

VB // OLESetData event is not supported. Use the DragEnter,DragLeave,DragOver, DragDrop ... events.

Syntax for OLESetData event, /COM version, on:

C#	private void OLESetData(object sender,
	AxEXSCHEDULELibIScheduleEvents_OLESetDataEvent e)
	{
	}
C++	void OnOLESetData(LPDISPATCH Data, short Format)

	{ }
C++	voidfastcall OLESetData(TObject *Sender,Exschedulelib_tlb::IExDataObject
Builder	*Data,short Format)
	{
	}

Delphi	
proceo begin end;	dure OLESetData(ASender: TObject; Data : IExDataObject;Format : Smallint);
Delphi 8 (.NET only)	procedure OLESetData(sender: System.Object; e: AxEXSCHEDULELibIScheduleEvents_OLESetDataEvent); begin end;
Powe	begin event OLESetData(oleobject Data,integer Format) end event OLESetData
VB.NET	Private Sub OLESetData(ByVal sender As System.Object, ByVal e As AxEXSCHEDULELibIScheduleEvents_OLESetDataEvent) Handles OLESetData End Sub
VB6	Private Sub OLESetData(ByVal Data As EXSCHEDULELibCtl.IExDataObject,ByVal Format As Integer) End Sub
VBA	Private Sub OLESetData(ByVal Data As Object,ByVal Format As Integer) End Sub
VFP	LPARAMETERS Data,Format
Xbas	PROCEDURE OnOLESetData(oSchedule,Data,Format) RETURN

Syntax for OLESetData event, /COM version ^(others), on:

Java...

<SCRIPT EVENT="OLESetData(Data,Format)" LANGUAGE="JScript"> </SCRIPT>

VBSc	<script language="VBScript"></script>
------	---------------------------------------

	Procedure OnComOLESetData Variant IIData Short IIEormat
Visual Data	Forward Send OnComOLESetData IIData IIFormat End_Procedure
Visual Objects	METHOD OCX_OLESetData(Data,Format) CLASS MainDialog RETURN NIL
Х++	<pre>// OLESetData event is not supported. Use the DragEnter,DragLeave,DragOver, DragDrop events.</pre>
XBasic	function OLESetData as v (Data as OLE::Exontrol.Schedule.1::IExDataObject,Format as N) end function
dBASE	function nativeObject_OLESetData(Data,Format) return

event OLEStartDrag (Data as ExDataObject, AllowedEffects as Long)

Occurs when the OLEDrag method is called.

Туре	Description
Data as <u>ExDataObject</u>	An ExDataObject object containing formats that the source will provide and, optionally, the data for those formats. If no data is contained in the ExDataObject, it is provided when the control calls the GetData method. The programmer should provide the values for this parameter in this event. The SetData and Clear methods cannot be used here.
AllowedEffects as Long	A long containing the effects that the source component supports. The possible values are listed in Settings. The programmer should provide the values for this parameter in this event

In the /NET Assembly, you have to use the DragEnter event as explained here:

https://www.exontrol.com/sg.jsp?content=support/faq/net/#dragdrop

The <u>Background(exScheduleOLEDropPosition</u>) property specifies the visual appearance of the line to be shown when the cursor is hovering the schedule part of the control, during an OLE drag and drop operation.

The settings for AllowEffects are:

- exOLEDropEffectNone (0), Drop target cannot accept the data, or the drop operation was cancelled
- exOLEDropEffectCopy (1), Drop results in a copy of data from the source to the target. The original data is unaltered by the drag operation.
- exOLEDropEffectMove (2), Drop results in data being moved from drag source to drop source. The drag source should remove the data from itself after the move.

The source component should logically Or together the supported values and places the result in the AllowedEffects parameter. The target component can use this value to determine the appropriate action (and what the appropriate user feedback should be). You may wish to defer putting data into the ExDataObject object until the target component requests it. This allows the source component to save time. If the user does not load any formats into the ExDataObject, then the drag/drop operation is canceled. Use <u>exCFFiles</u> and <u>Files</u> property to add files to the drag and drop data object.

The idea of drag and drop in exSchedule control is the same as in other controls. To start accepting drag and drop sources the exSchedule control MUST have the <u>OLEDropMode</u>

property to exOLEDropManual. Once that is is set, the exSchedule starts accepting any drag and drop sources.

The first step is if you want to be able to drag items from your exSchedule control to other controls the idea is to handle the OLE_StartDrag event. The event passes an object ExDataObject (Data) as argument. The Data and AllowedEffects can be changed only in the OLEStartDrag event. The OLE_StartDrag event is fired when user is about to drag items from the control. The AllowedEffect parameter and <u>SetData</u> property must be set to continue drag and drop operation, as in the following sample:

Syntax for OLEStartDrag event, /NET version, on:

C# // OLEStartDrag event is not supported. Use the DragEnter,DragLeave,DragOver, DragDrop ... events.

// OLEStartDrag event is not supported. Use the DragEnter,DragLeave,DragOver, DragDrop ... events.

Syntax for OLEStartDrag event, /COM version, on:

VB

C#	private void OLEStartDrag(object sender,
	AxEXSCHEDULELibIScheduleEvents_OLEStartDragEvent e)
	{
	}

C++	void OnOLEStartDrag(LPDISPATCH Data,long FAR* AllowedEffects)
	{
	}

C++ Builder	<pre>voidfastcall OLEStartDrag(TObject *Sender,Exschedulelib_tlb::IExDataObject *Data,long * AllowedEffects) { }</pre>
Delphi	procedure OLEStartDrag(ASender: TObject; Data : IExDataObject;var AllowedEffects : Integer); begin end;



procee AxEXS begin end;	dure OLEStartDrag(sender: System.Object; e: CHEDULELibIScheduleEvents_OLEStartDragEvent);
Powe	begin event OLEStartDrag(oleobject Data,long AllowedEffects) end event OLEStartDrag
VB.NET	Private Sub OLEStartDrag(ByVal sender As System.Object, ByVal e As AxEXSCHEDULELibIScheduleEvents_OLEStartDragEvent) Handles OLEStartDrag End Sub
VB6	Private Sub OLEStartDrag(ByVal Data As EXSCHEDULELibCtl.IExDataObject,AllowedEffects As Long) End Sub
VBA	Private Sub OLEStartDrag(ByVal Data As Object,AllowedEffects As Long) End Sub
VFP	LPARAMETERS Data, Allowed Effects
Xbas	PROCEDURE OnOLEStartDrag(oSchedule,Data,AllowedEffects) RETURN
Syntax	for OLEStartDrag event, / COM version ^(others) , on:
Java	<script event="OLEStartDrag(Data,AllowedEffects)" language="JScript"> </script>

VBSc	<script language="VBScript"></th></tr><tr><th></th><th>Function OLEStartDrag(Data,AllowedEffects)</th></tr><tr><th></th><th>End Function</th></tr><tr><th></th><th></script>

visuai	
Data	

Procedure OnComOLEStartDrag Variant IIData Integer IIAllowedEffects Forward Send OnComOLEStartDrag IIData IIAllowedEffects End_Procedure				
Visual Objects	METHOD OCX_OLEStartDrag(Data,AllowedEffects) CLASS MainDialog RETURN NIL			
X++	<pre>// OLEStartDrag event is not supported. Use the DragEnter,DragLeave,DragOver, DragDrop events.</pre>			
XBasic	function OLEStartDrag as v (Data as OLE::Exontrol.Schedule.1::IExDataObject,AllowedEffects as N) end function			
dBASE	function nativeObject_OLEStartDrag(Data,AllowedEffects) return			

The following VB sample collects the focused/selected events and start dragging them (you can drop them to a WinWord, Excel application for instance):

```
Private Sub Schedule1_OLEStartDrag(ByVal Data As EXSCHEDULELibCtl.IExDataObject,
AllowedEffects As Long)
With Schedule1
Dim e As EXSCHEDULELibCtl.Event
Set e = .EventFromPoint(-1, -1)
If Not e Is Nothing Then
Dim sDragDropText As String
If Not (e.Selected) Then
sDragDropText = sDragDropText & "Start: " & e.Start & ", End: " & e.End &
vbCrLf
Else
Dim s As Variant
For Each s In .Selection
sDragDropText = sDragDropText & "Start: " & s.Start & ", End: " & s.End &
vbCrLf
```

```
Next
End If
AllowedEffects = 1
Data.SetData sDragDropText
End If
End With
End Sub
```

event PictureClick (Key as String)

Occurs when the user clicks a picture within an event (Event.Pictures/ExtraPictures).

Туре	Description
Key as String	A String expression that specifies the picture being clicked. The Key parameter indicates the Key being used when Pictures.Add method has been called.

The PictureClick event is fired once the user clicks a picture inside an event. Use the <u>Add</u> method of the <u>ExPictures</u> collection to add new images to the control. Use the <u>Images</u> method to add a list of icons to be used in the control. The <u>Pictures</u> or <u>ExtraPictures</u> property of the <u>Event</u> object indicates the list of pictures to be displayed in the Event object. The Pictures or ExtraPictures property may display a single or multiple pictures on different levels, using separators like , multiple elements in the same line, or / for a new line/level. You can use the <u>ShowHandCursor</u> property to specify whether the hand cursor is shown when the cursor hovers the picture. Also, you can display the images/icons on the <u>LongLabel/ExtraLabel</u> using the HTML tag that can be include to a <a> HTML tag, like <a pic1>pic1. You can use the <u>ShowtLabel</u> is displayed, but it can not display HTML tags. In other words, if the event's body can not display the picture to adjust the size of the displaying picture.

The <u>PictureFromPoint</u> property indicates the key of the picture from the cursor or empty string if no picture has been found. The <u>EventFromPoint</u> property may be used to get the Event object from the cursor.

Syntax for PictureClick event, /NET version, on:

C#	private void PictureClick(object sender, string Key)
	{
	}

Private Sub PictureClick(ByVal sender As System.Object,ByVal Key As String)
 Handles PictureClick
 End Sub

Syntax for PictureClick event, /COM version, on:

private void PictureClick(object sender,
 AxEXSCHEDULELib._IScheduleEvents_PictureClickEvent e)

	{ }
C++	void OnPictureClick(LPCTSTR Key) { }
C++ Builder	voidfastcall PictureClick(TObject *Sender,BSTR Key) { }
Delphi	procedure PictureClick(ASender: TObject; Key : WideString); begin end;
Delphi 8 (.NET only)	procedure PictureClick(sender: System.Object; e: AxEXSCHEDULELibIScheduleEvents_PictureClickEvent); begin end:
Powe	begin event PictureClick(string Key) end event PictureClick
Powe VB.NET	begin event PictureClick(string Key) end event PictureClick Private Sub PictureClick(ByVal sender As System.Object, ByVal e As AxEXSCHEDULELibIScheduleEvents_PictureClickEvent) Handles PictureClick End Sub
Powe VB.NET VB6	begin event PictureClick(string Key) end event PictureClick Private Sub PictureClick(ByVal sender As System.Object, ByVal e As AxEXSCHEDULELibIScheduleEvents_PictureClickEvent) Handles PictureClick End Sub Private Sub PictureClick(ByVal Key As String) End Sub
Powe VB.NET VB6	begin event PictureClick(string Key) end event PictureClick Private Sub PictureClick(ByVal sender As System.Object, ByVal e As AxEXSCHEDULELibIScheduleEvents_PictureClickEvent) Handles PictureClick End Sub Private Sub PictureClick(ByVal Key As String) End Sub Private Sub PictureClick(ByVal Key As String) End Sub
Powe VB.NET VB6 VB6 VBA	 begin event PictureClick(string Key) end event PictureClick Private Sub PictureClick(ByVal sender As System.Object, ByVal e As AxEXSCHEDULELibIScheduleEvents_PictureClickEvent) Handles PictureClick End Sub Private Sub PictureClick(ByVal Key As String) End Sub Private Sub PictureClick(ByVal Key As String) End Sub Private Sub PictureClick(ByVal Key As String) End Sub LPARAMETERS Key

Syntax for PictureClick event, /COM version ^(others), on:

Java	<script event="PictureClick(Key)" language="JScript"> </script>
VBSc	<script language="VBScript"> Function PictureClick(Key) End Function </script>
Visual Data	Procedure OnComPictureClick String IIKey Forward Send OnComPictureClick IIKey End_Procedure
Visual Objects	METHOD OCX_PictureClick(Key) CLASS MainDialog RETURN NIL
X++	void onEvent_PictureClick(str _Key) { }
XBasic	function PictureClick as v (Key as C) end function
dBASE	function nativeObject_PictureClick(Key) return

The following samples displays a message once a picture is clicked:

VBA (MS Access, Excell...)



```
End Sub
With Schedule1
  .BeginUpdate
  .OnResizeControl = 3073
  .Calendar.Selection = #5/24/2012#
  With .Pictures
    With .Add("pic1","c:\exontrol\images\card.png")
      .Width = 48
      .Height = 48
    End With
    With .Add("pic2","c:\exontrol\images\diary.png")
      .Width = 48
      .Height = 48
    End With
  End With
  With Events
    .Add(#5/24/2012 8:45:00 AM#,#5/24/2012 2:30:00 PM#).Pictures = "pic1"
    With .Add(#5/24/2012 9:45:00 AM#,#5/24/2012 3:45:00 PM#)
      .ExtraPictures = "pic1,pic2"
      .ExtraPicturesAlign = 34
    End With
  End With
  .EndUpdate
End With
```

VB6

' PictureClick event - Occurs when the user clicks a picture within an event (Event.Pictures/ExtraPictures).

Private Sub Schedule1_PictureClick(ByVal Key As String) With Schedule1 Debug.Print(Key) End With End Sub

With Schedule1

```
.BeginUpdate
  .OnResizeControl = OnResizeControlEnum.exResizePanelRight Or
OnResizeControlEnum.exCalendarFit Or OnResizeControlEnum.exCalendarAutoHide
  .Calendar.Selection = \#5/24/2012\#
  With .Pictures
    With .Add("pic1","c:\exontrol\images\card.png")
      .Width = 48
      .Height = 48
    End With
    With .Add("pic2","c:\exontrol\images\diary.png")
      .Width = 48
      .Height = 48
    End With
  End With
  With Events
    .Add(#5/24/2012 8:45:00 AM#,#5/24/2012 2:30:00 PM#).Pictures = "pic1"
    With .Add(#5/24/2012 9:45:00 AM#,#5/24/2012 3:45:00 PM#)
      .ExtraPictures = "pic1,pic2"
      .ExtraPicturesAlign = exBottomRight
    End With
  End With
  .EndUpdate
End With
```

VB.NET

' PictureClick event - Occurs when the user clicks a picture within an event (Event.Pictures/ExtraPictures).

Private Sub Exschedule1_PictureClick(ByVal sender As System.Object,ByVal Key As String) Handles Exschedule1.PictureClick

With Exschedule1

Debug.Print(Key)

End With

End Sub

With Exschedule1 .BeginUpdate()

```
.OnResizeControl =
exontrol.EXSCHEDULELib.OnResizeControlEnum.exResizePanelRight Or
exontrol.EXSCHEDULELib.OnResizeControlEnum.exCalendarFit Or
exontrol.EXSCHEDULELib.OnResizeControlEnum.exCalendarAutoHide
  .Calendar.Selection = #5/24/2012#
  With .Pictures
    With .Add("pic1","c:\exontrol\images\card.png")
      .Width = 48
      .Height = 48
    End With
    With .Add("pic2","c:\exontrol\images\diary.png")
      .Width = 48
      .Height = 48
    End With
  End With
  With Events
    .Add(#5/24/2012 8:45:00 AM#,#5/24/2012 2:30:00 PM#).Pictures = "pic1"
    With .Add(#5/24/2012 9:45:00 AM#,#5/24/2012 3:45:00 PM#)
      .ExtraPictures = "pic1,pic2"
      .ExtraPicturesAlign =
exontrol.EXSCHEDULELib.ContentAlignmentEnum.exBottomRight
    End With
  End With
  .EndUpdate()
End With
```

VB.NET for /COM

' PictureClick event - Occurs when the user clicks a picture within an event (Event.Pictures/ExtraPictures).

Private Sub AxSchedule1_PictureClick(ByVal sender As System.Object, ByVal e As AxEXSCHEDULELib._IScheduleEvents_PictureClickEvent) Handles AxSchedule1.PictureClick With AxSchedule1 Debug.Print(e.key) End With End Sub

```
With AxSchedule1
  .BeginUpdate()
  .OnResizeControl = EXSCHEDULELib.OnResizeControlEnum.exResizePanelRight Or
EXSCHEDULELib.OnResizeControlEnum.exCalendarFit Or
EXSCHEDULELib.OnResizeControlEnum.exCalendarAutoHide
  .Calendar.Selection = #5/24/2012#
  With .Pictures
    With .Add("pic1","c:\exontrol\images\card.png")
      .Width = 48
      .Height = 48
    End With
    With .Add("pic2","c:\exontrol\images\diary.png")
      .Width = 48
      .Height = 48
    End With
  End With
  With .Events
    .Add(#5/24/2012 8:45:00 AM#,#5/24/2012 2:30:00 PM#).Pictures = "pic1"
    With .Add(#5/24/2012 9:45:00 AM#,#5/24/2012 3:45:00 PM#)
      .ExtraPictures = "pic1,pic2"
      .ExtraPicturesAlign = EXSCHEDULELib.ContentAlignmentEnum.exBottomRight
    End With
  End With
  .EndUpdate()
End With
```

C++

```
// PictureClick event - Occurs when the user clicks a picture within an event (
Event.Pictures/ExtraPictures ).
void OnPictureClickSchedule1(LPCTSTR Key)
{
    /*
    Copy and paste the following directives to your header file as
    it defines the namespace 'EXSCHEDULELib' for the library: 'ExSchedule 1.0
Control Library'
```
```
#import <ExSchedule.dll>
    using namespace EXSCHEDULELib;
  */
  EXSCHEDULELib::ISchedulePtr spSchedule1 = GetDlgItem(IDC_SCHEDULE1)-
>GetControlUnknown();
  OutputDebugStringW( L"Key" );
}
EXSCHEDULELib::ISchedulePtr spSchedule1 = GetDlgItem(IDC_SCHEDULE1)-
>GetControlUnknown();
spSchedule1->BeginUpdate();
spSchedule1-
> PutOnResizeControl(EXSCHEDULELib::OnResizeControlEnum(EXSCHEDULELib::exResizeControlEnum)
| EXSCHEDULELib::exCalendarFit | EXSCHEDULELib::exCalendarAutoHide));
spSchedule1->GetCalendar()->PutSelection("5/24/2012");
EXSCHEDULELib::IExPicturesPtr var_ExPictures = spSchedule1->GetPictures();
  EXSCHEDULELib::IExPicturePtr var ExPicture = var ExPictures-
>Add(L"pic1","c:\\exontrol\\images\\card.png");
    var_ExPicture->PutWidth(48);
    var_ExPicture->PutHeight(48);
  EXSCHEDULELib::IExPicturePtr var_ExPicture1 = var_ExPictures-
>Add(L"pic2","c:\\exontrol\\images\\diary.png");
    var ExPicture1->PutWidth(48);
    var_ExPicture1->PutHeight(48);
EXSCHEDULELib::IEventsPtr var_Events = spSchedule1->GetEvents();
  var_Events->Add("5/24/2012 8:45:00 AM","5/24/2012 2:30:00 PM")-
> PutPictures(L"pic1");
  EXSCHEDULELib::IEventPtr var_Event = var_Events->Add("5/24/2012 9:45:00
AM","5/24/2012 3:45:00 PM");
    var_Event->PutExtraPictures(L"pic1,pic2");
    var_Event->PutExtraPicturesAlign(EXSCHEDULELib::exBottomRight);
spSchedule1->EndUpdate();
```

C++ Builder

// PictureClick event - Occurs when the user clicks a picture within an event (

```
Event.Pictures/ExtraPictures ).
void __fastcall TForm1::Schedule1PictureClick(TObject *Sender,BSTR Key)
  OutputDebugString(L"Key");
}
Schedule1->BeginUpdate();
Schedule1->OnResizeControl =
Exschedulelib_tlb::OnResizeControlEnum::exResizePanelRight |
Exschedulelib_tlb::OnResizeControlEnum::exCalendarFit |
Exschedulelib_tlb::OnResizeControlEnum::exCalendarAutoHide;
Schedule1->Calendar->set_Selection(TVariant(TDateTime(2012,5,24).operator
double()));
Exschedulelib_tlb::IExPicturesPtr var_ExPictures = Schedule1->Pictures;
  Exschedulelib_tlb::IExPicturePtr var_ExPicture = var_ExPictures-
>Add(L"pic1",TVariant("c:\\exontrol\\images\\card.png"));
    var ExPicture->Width = 48;
    var_ExPicture -> Height = 48;
  Exschedulelib_tlb::IExPicturePtr var_ExPicture1 = var_ExPictures-
>Add(L"pic2",TVariant("c:\\exontrol\\images\\diary.png"));
    var_ExPicture1->Width = 48;
    var_ExPicture1 -> Height = 48;
Exschedulelib_tlb::IEventsPtr var_Events = Schedule1->Events;
  var_Events->Add(TVariant(TDateTime(2012,5,24,8,45,00,0).operator
double()),TVariant(TDateTime(2012,5,24,14,30,00,0).operator double()))->Pictures =
L"pic1";
  Exschedulelib_tlb::IEventPtr var_Event = var_Events-
>Add(TVariant(TDateTime(2012,5,24,9,45,00,0).operator
double()),TVariant(TDateTime(2012,5,24,15,45,00,0).operator double()));
    var_Event->ExtraPictures = L"pic1,pic2";
    var_Event->ExtraPicturesAlign =
Exschedulelib_tlb::ContentAlignmentEnum::exBottomRight;
```

```
Schedule1->EndUpdate();
```

```
// PictureClick event - Occurs when the user clicks a picture within an event (
Event.Pictures/ExtraPictures ).
private void exschedule1_PictureClick(object sender, string Key)
  System.Diagnostics.Debug.Print( Key.ToString() );
}
//this.exschedule1.PictureClick += new
exontrol.EXSCHEDULELib.exg2antt.PictureClickEventHandler(this.exschedule1_F
exschedule1.BeginUpdate();
exschedule1.OnResizeControl =
exontrol.EXSCHEDULELib.OnResizeControlEnum.exResizePanelRight |
exontrol.EXSCHEDULELib.OnResizeControlEnum.exCalendarFit |
exontrol.EXSCHEDULELib.OnResizeControlEnum.exCalendarAutoHide;
exschedule1.Calendar.Selection =
Convert.ToDateTime("5/24/2012",System.Globalization.CultureInfo.GetCultureInfo("en-
US"));
exontrol.EXSCHEDULELib.ExPictures var_ExPictures = exschedule1.Pictures;
  exontrol.EXSCHEDULELib.ExPicture var_ExPicture =
var_ExPictures.Add("pic1","c:\\exontrol\\images\\card.png");
    var_ExPicture.Width = 48;
    var_ExPicture.Height = 48;
  exontrol.EXSCHEDULELib.ExPicture var_ExPicture1 =
var_ExPictures.Add("pic2","c:\\exontrol\\images\\diary.png");
    var_ExPicture1.Width = 48;
    var_ExPicture1.Height = 48;
exontrol.EXSCHEDULELib.Events var_Events = exschedule1.Events;
  var_Events.Add(Convert.ToDateTime("5/24/2012 8:45:00
AM", System. Globalization. CultureInfo. GetCultureInfo("en-
US")),Convert.ToDateTime("5/24/2012 2:30:00
PM",System.Globalization.CultureInfo.GetCultureInfo("en-US"))).Pictures = "pic1";
  exontrol.EXSCHEDULELib.Event var Event =
var_Events.Add(Convert.ToDateTime("5/24/2012 9:45:00
AM", System. Globalization. CultureInfo. GetCultureInfo("en-
US")),Convert.ToDateTime("5/24/2012 3:45:00
PM",System.Globalization.CultureInfo.GetCultureInfo("en-US")));
```

```
var_Event.ExtraPictures = "pic1,pic2";
var_Event.ExtraPicturesAlign =
exontrol.EXSCHEDULELib.ContentAlignmentEnum.exBottomRight;
exschedule1.EndUpdate();
```

JavaScript

```
<SCRIPT FOR="Schedule1" EVENT="PictureClick(Key)" LANGUAGE="JScript">
  alert(Key);
</SCRIPT>
<OBJECT classid = "clsid:9B09E13D-7A88-4299-9DBE-383380435377"
id="Schedule1"></OBJECT>
<SCRIPT LANGUAGE="JScript">
  Schedule1.BeginUpdate();
  Schedule1.OnResizeControl = 3073;
  Schedule1.Calendar.Selection = "5/24/2012";
  var var ExPictures = Schedule1.Pictures;
    var var_ExPicture = var_ExPictures.Add("pic1","c:\\exontrol\\images\\card.png");
      var ExPicture.Width = 48;
      var_ExPicture.Height = 48;
    var var_ExPicture1 =
var_ExPictures.Add("pic2","c:\\exontrol\\images\\diary.png");
      var_ExPicture1.Width = 48;
      var_ExPicture1.Height = 48;
  var var Events = Schedule1.Events;
    var_Events.Add("5/24/2012 8:45:00 AM","5/24/2012 2:30:00 PM").Pictures =
"pic1";
    var var_Event = var_Events.Add("5/24/2012 9:45:00 AM","5/24/2012 3:45:00
PM");
      var_Event.ExtraPictures = "pic1,pic2";
      var_Event.ExtraPicturesAlign = 34;
  Schedule1.EndUpdate();
</SCRIPT>
```

// PictureClick event - Occurs when the user clicks a picture within an event (**Event.Pictures/ExtraPictures**). private void axSchedule1_PictureClick(object sender, AxEXSCHEDULELib._IScheduleEvents_PictureClickEvent e) { System.Diagnostics.Debug.Print(e.key.ToString()); } //this.axSchedule1.PictureClick += new AxEXSCHEDULELib. IScheduleEvents_PictureClickEventHandler(this.axSchedule1 axSchedule1.BeginUpdate(); axSchedule1.OnResizeControl = EXSCHEDULELib.OnResizeControlEnum.exResizePanelRight | EXSCHEDULELib.OnResizeControlEnum.exCalendarFit | EXSCHEDULELib.OnResizeControlEnum.exCalendarAutoHide; axSchedule1.Calendar.Selection = Convert.ToDateTime("5/24/2012",System.Globalization.CultureInfo.GetCultureInfo("en-**US**")); EXSCHEDULELib.ExPictures var_ExPictures = axSchedule1.Pictures; EXSCHEDULELib.ExPicture var_ExPicture = var_ExPictures.Add("pic1","c:\\exontrol\\images\\card.png"); var_ExPicture.Width = 48; var_ExPicture.Height = 48; EXSCHEDULELib.ExPicture var_ExPicture1 = var_ExPictures.Add("pic2","c:\\exontrol\\images\\diary.png"); var_ExPicture1.Width = 48; $var_ExPicture1.Height = 48;$ EXSCHEDULELib.Events var_Events = axSchedule1.Events; var_Events.Add(Convert.ToDateTime("5/24/2012 8:45:00 AM", System. Globalization. CultureInfo. GetCultureInfo("en-US")),Convert.ToDateTime("5/24/2012 2:30:00 PM",System.Globalization.CultureInfo.GetCultureInfo("en-US"))).Pictures = "pic1"; EXSCHEDULELib.Event var Event = var_Events.Add(Convert.ToDateTime("5/24/2012 9:45:00 AM", System. Globalization. CultureInfo. GetCultureInfo("en-US")),Convert.ToDateTime("5/24/2012 3:45:00

```
PM",System.Globalization.CultureInfo.GetCultureInfo("en-US")));
    var_Event.ExtraPictures = "pic1,pic2";
    var_Event.ExtraPicturesAlign =
    EXSCHEDULELib.ContentAlignmentEnum.exBottomRight;
    axSchedule1.EndUpdate();
```

X++ (Dynamics Ax 2009)

```
// PictureClick event - Occurs when the user clicks a picture within an event (
Event.Pictures/ExtraPictures ).
void onEvent_PictureClick(str _Key)
{
  print(_Key);
}
public void init()
  COM com_Event,com_Events,com_ExPicture,com_ExPicture1,com_ExPictures;
  anytype var_Event,var_Events,var_ExPicture,var_ExPicture1,var_ExPictures;
  super();
  exschedule1.BeginUpdate();
  exschedule1.OnResizeControl(3073/*exResizePanelRight | exCalendarFit |
exCalendarAutoHide*/);
exschedule1.Calendar().Selection(COMVariant::createFromDate(str2Date("5/24/2012",2
  var_ExPictures = exschedule1.Pictures(); com_ExPictures = var_ExPictures;
    var_ExPicture = com_ExPictures.Add("pic1","c:\\exontrol\\images\\card.png");
com_ExPicture = var_ExPicture;
      com_ExPicture.Width(48);
      com_ExPicture.Height(48);
    var_ExPicture1 = com_ExPictures.Add("pic2","c:\\exontrol\\images\\diary.png");
```

```
com_ExPicture1 = var_ExPicture1;
      com ExPicture1.Width(48);
      com_ExPicture1.Height(48);
  var_Events = exschedule1.Events(); com_Events = var_Events;
    var Event =
COM::createFromObject(com_Events.Add(COMVariant::createFromUtcDateTime(str2Da
8:45:00",213)),COMVariant::createFromUtcDateTime(str2Datetime("5/24/2012
14:30:00",213)))); com_Event = var_Event;
    com_Event.Pictures("pic1");
    var Event =
com_Events.Add(COMVariant::createFromUtcDateTime(str2Datetime("5/24/2012
9:45:00",213)),COMVariant::createFromUtcDateTime(str2Datetime("5/24/2012
15:45:00",213))); com_Event = var_Event;
      com_Event.ExtraPictures("pic1,pic2");
      com_Event.ExtraPicturesAlign(34/*exBottomRight*/);
  exschedule1.EndUpdate();
```

Delphi 8 (.NET only)

// PictureClick event - Occurs when the user clicks a picture within an event (Event.Pictures/ExtraPictures).

```
procedure TWinForm1.AxSchedule1_PictureClick(sender: System.Object; e:
AxEXSCHEDULELib._IScheduleEvents_PictureClickEvent);
begin
with AxSchedule1 do
begin
OutputDebugString( e.key );
end
end;
with AxSchedule1 do
begin
BeginUpdate();
OnResizeControl :=
Integer(EXSCHEDULELib.OnResizeControlEnum.exResizePanelRight) Or
Integer(EXSCHEDULELib.OnResizeControlEnum.exCalendarFit) Or
```

```
Integer(EXSCHEDULELib.OnResizeControlEnum.exCalendarAutoHide);
  Calendar.Selection := '5/24/2012';
  with Pictures do
  begin
    with Add('pic1','c:\exontrol\images\card.png') do
    begin
      Width := 48;
      Height := 48;
    end;
    with Add('pic2','c:\exontrol\images\diary.png') do
    begin
      Width := 48:
      Height := 48;
    end;
  end;
  with Events do
  begin
    Add('5/24/2012 8:45:00 AM','5/24/2012 2:30:00 PM').Pictures := 'pic1';
    with Add('5/24/2012 9:45:00 AM','5/24/2012 3:45:00 PM') do
    begin
      ExtraPictures := 'pic1,pic2';
      ExtraPicturesAlign := EXSCHEDULELib.ContentAlignmentEnum.exBottomRight;
    end;
  end;
  EndUpdate();
end
```

Delphi (standard)

// PictureClick event - Occurs when the user clicks a picture within an event (Event.Pictures/ExtraPictures).

```
procedure TForm1.Schedule1PictureClick(ASender: TObject; Key : WideString);
begin
with Schedule1 do
begin
OutputDebugString( Key );
end
```

end; with Schedule1 do begin BeginUpdate(); OnResizeControl := Integer(EXSCHEDULELib_TLB.exResizePanelRight) Or Integer(EXSCHEDULELib_TLB.exCalendarFit) Or Integer(EXSCHEDULELib_TLB.exCalendarAutoHide); Calendar.Selection := '5/24/2012'; with **Pictures** do begin with Add('pic1','c:\exontrol\images\card.png') do begin Width := 48; Height := 48;end; with Add('pic2','c:\exontrol\images\diary.png') do begin Width := 48; Height := 48; end; end: with Events do begin Add('5/24/2012 8:45:00 AM','5/24/2012 2:30:00 PM').Pictures := 'pic1'; with Add('5/24/2012 9:45:00 AM','5/24/2012 3:45:00 PM') do begin **ExtraPictures** := 'pic1,pic2'; ExtraPicturesAlign := EXSCHEDULELib_TLB.exBottomRight; end; end; EndUpdate(); end

VFP

```
Event.Pictures/ExtraPictures ). ***
LPARAMETERS Key
  with thisform.Schedule1
    DEBUGOUT( Key )
  endwith
with thisform.Schedule1
  .BeginUpdate
  .OnResizeControl = 3073
  .Calendar.Selection = ^{2012-5-24}
  with .Pictures
    with .Add("pic1","c:\exontrol\images\card.png")
      .Width = 48
      .Height = 48
    endwith
    with .Add("pic2","c:\exontrol\images\diary.png")
      .Width = 48
      .Height = 48
    endwith
  endwith
  with .Events
    .Add({^2012-5-24 8:45:00},{^2012-5-24 14:30:00}).Pictures = "pic1"
    with .Add({^2012-5-24 9:45:00},{^2012-5-24 15:45:00})
      .ExtraPictures = "pic1,pic2"
      .ExtraPicturesAlign = 34
    endwith
  endwith
  .EndUpdate
```

```
endwith
```

dBASE Plus

```
/*
with (this.ACTIVEX1.nativeObject)
PictureClick = class::nativeObject_PictureClick
endwith
*/
```

```
// Occurs when the user clicks a picture within an event (
Event.Pictures/ExtraPictures ).
```

```
function nativeObject_PictureClick(Key)
local oSchedule
oSchedule = form.Activex1.nativeObject
? Str(Key)
return
```

```
local
oSchedule,var_Event,var_Event1,var_Events,var_ExPicture,var_ExPicture1,var_ExPictures
```

```
oSchedule = form.Activex1.nativeObject
oSchedule.BeginUpdate()
oSchedule.OnResizeControl = 3073 /*exResizePanelRight | exCalendarFit |
exCalendarAutoHide*/
oSchedule.Calendar.Selection = "05/24/2012"
var ExPictures = oSchedule.Pictures
  var_ExPicture = var_ExPictures.Add("pic1","c:\exontrol\images\card.png")
    var ExPicture.Width = 48
    var_ExPicture.Height = 48
  var_ExPicture1 = var_ExPictures.Add("pic2","c:\exontrol\images\diary.png")
    var ExPicture1.Width = 48
    var_ExPicture1.Height = 48
var_Events = oSchedule.Events
  // var_Events.Add("05/24/2012 08:45:00","05/24/2012 14:30:00").Pictures
= "pic1"
  var_Event = var_Events.Add("05/24/2012 08:45:00","05/24/2012 14:30:00")
  with (oSchedule)
    TemplateDef = [Dim var_Event]
    TemplateDef = var_Event
    Template = [var_Event.Pictures = "pic1"]
  endwith
  var_Event1 = var_Events.Add("05/24/2012 09:45:00","05/24/2012 15:45:00")
    var_Event1.ExtraPictures = "pic1,pic2"
    var_Event1.ExtraPicturesAlign = 34
oSchedule.EndUpdate()
```

XBasic (Alpha Five)

```
' Occurs when the user clicks a picture within an event (
Event.Pictures/ExtraPictures ).
function PictureClick as v (Key as C)
  Dim oSchedule as P
  oSchedule = topparent:CONTROL_ACTIVEX1.activex
  ? Key
end function
Dim oSchedule as P
Dim var Event as P
Dim var Event1 as P
Dim var Events as P
Dim var ExPicture as P
Dim var ExPicture1 as P
Dim var_ExPictures as P
oSchedule = topparent:CONTROL_ACTIVEX1.activex
oSchedule.BeginUpdate()
oSchedule.OnResizeControl = 3073 'exResizePanelRight + exCalendarFit +
exCalendarAutoHide
oSchedule.Calendar.Selection = {05/24/2012}
var ExPictures = oSchedule.Pictures
  var_ExPicture = var_ExPictures.Add("pic1","c:\exontrol\images\card.png")
    var ExPicture.Width = 48
    var_ExPicture.Height = 48
  var_ExPicture1 = var_ExPictures.Add("pic2","c:\exontrol\images\diary.png")
    var ExPicture1.Width = 48
    var_ExPicture1.Height = 48
var_Events = oSchedule.Events
  ' var Events.Add({05/24/2012 08:45:00},{05/24/2012 14:30:00}).Pictures =
"pic1"
```

```
var_Event = var_Events.Add({05/24/2012 08:45:00},{05/24/2012 14:30:00})
oSchedule.TemplateDef = "Dim var_Event"
```

```
oSchedule.Template = "var_Event.Pictures = \"pic1\""
var_Event1 = var_Events.Add({05/24/2012 09:45:00},{05/24/2012 15:45:00})
var_Event1.ExtraPictures = "pic1,pic2"
var_Event1.ExtraPicturesAlign = 34
oSchedule.EndUpdate()
```

Visual Objects

oSchedule.TemplateDef = var_Event

```
METHOD OCX_Exontrol1PictureClick(Key) CLASS MainDialog
  // PictureClick event - Occurs when the user clicks a picture within an event
(Event.Pictures/ExtraPictures).
  OutputDebugString(String2Psz(AsString(Key)))
RETURN NIL
local var Event as IEvent
local var Events as IEvents
local var_ExPicture,var_ExPicture1 as IExPicture
local var ExPictures as IExPictures
oDCOCX_Exontrol1:BeginUpdate()
oDCOCX_Exontrol1:OnResizeControl := exResizePanelRight | exCalendarFit |
exCalendarAutoHide
oDCOCX_Exontrol1:Calendar:Selection := SToD("20120524")
var ExPictures := oDCOCX Exontrol1:Pictures
  var_ExPicture := var_ExPictures:Add("pic1","c:\exontrol\images\card.png")
    var ExPicture:Width := 48
    var_ExPicture:Height := 48
  var_ExPicture1 := var_ExPictures:Add("pic2","c:\exontrol\images\diary.png")
    var ExPicture1:Width := 48
    var_ExPicture1:Height := 48
var Events := oDCOCX Exontrol1:Events
  var_Events:Add(SToD("20120524 08:45:00"),SToD("20120524 14:30:00")):Pictures
:= "pic1"
  var_Event := var_Events:Add(SToD("20120524 09:45:00"),SToD("20120524
```

```
15:45:00"))
var_Event:ExtraPictures := "pic1,pic2"
var_Event:ExtraPicturesAlign := exBottomRight
oDCOCX_Exontrol1:EndUpdate()
```

PowerBuilder

```
/*begin event PictureClick(string Key) - Occurs when the user clicks a picture within an
event (Event.Pictures/ExtraPictures).*/
  OleObject oSchedule
  oSchedule = ole_1.Object
  MessageBox("Information",string( String(Key) ))
/*end event PictureClick*/
OleObject oSchedule,var_Event,var_Events,var_ExPicture,var_ExPicture1,var_ExPictures
oSchedule = ole_1.Object
oSchedule.BeginUpdate()
oSchedule.OnResizeControl = 3073 /*exResizePanelRight | exCalendarFit |
exCalendarAutoHide*/
oSchedule.Calendar.Selection = 2012-05-24
var ExPictures = oSchedule.Pictures
  var_ExPicture = var_ExPictures.Add("pic1","c:\exontrol\images\card.png")
    var ExPicture.Width = 48
    var_ExPicture.Height = 48
  var_ExPicture1 = var_ExPictures.Add("pic2","c:\exontrol\images\diary.png")
    var ExPicture1.Width = 48
    var_ExPicture1.Height = 48
var Events = oSchedule.Events
  var_Events.Add(DateTime(2012-05-24,08:45:00),DateTime(2012-05-
24,14:30:00)).Pictures = "pic1"
  var_Event = var_Events.Add(DateTime(2012-05-24,09:45:00),DateTime(2012-05-
24,15:45:00))
    var_Event.ExtraPictures = "pic1,pic2"
```

var_Event.ExtraPicturesAlign = 34 oSchedule.EndUpdate()

event RClick ()

Occurs once the user right clicks the control.

Туре

Description

The RClick event notifies your application once the user presses and releases the right mouse button over the control. Unlike RClick event, the <u>Click</u> event occurs once the user left clicks the control. The RClick event does not occur, if the user presses the right mouse button, drag to a new position and releases the button. By default, the control does nothing if the user right clicks the control. You can handle the RCLick event to provide a popup context support. Use a <u>MouseDown</u> or <u>MouseUp</u> event procedure to specify actions that will occur when a mouse button is pressed or released. Unlike the Click and <u>DblClick</u> 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.

During RClick event you can use the ... FromPoint(-1,-1) properties to get UI elements of the scheduler from the cursor as follows:

- Calendar. <u>DateFromPoint</u> property gets the date from the cursor in the Calendar panel.
- <u>AnchorFromPoint</u> property gets the anchor (<a>) element from the cursor.
- <u>DateTimeFromPoint</u> property gets the date/time from the cursor, in the Schedule panel.
- <u>TimeFromPoint</u> property gets the time from the cursor, in the Schedule panel.
- EventFromPoint property gets the Event object from the cursor.
- <u>GroupFromPoint</u> property retrieves the <u>Group</u> object from the cursor. Use the GroupHeaderFromPoint property to the Group object when cursor hovers the group's header.
- <u>GroupHeaderFromPoint</u> property retrieves the <u>Group</u> object from the cursor, when the cursor hovers just the group's header.
- <u>MarkTimeFromPoint</u> property gets the <u>MarkTime</u> object from the cursor. A MarkTime object is identified by a date/time where it is displaying.
- <u>MarkZoneFromPoint</u> property gets the <u>MarkZone</u> object from the cursor. A MarkZone object is identified by a starting date/time and ending date/time where it is displaying.
- <u>NonworkingTimeFromPoint</u> property indicates the <u>NonworkingTime</u> object from the cursor.
- <u>PictureFromPoint</u> property indicates the key of the picture from the cursor.
- <u>TimeScaleFromPoint</u> property returns the <u>TimeScale</u> object from the cursor.

All ... FromPoint properties that returns an Object, may return Nothing, Empty or NULL, if no object is found.

Syntax for RClick event, /NET version, on:

private void RClick(object sender)

	{ }
VB	Private Sub RClick(ByVal sender As System.Object) Handles RClick End Sub
Syntax f	or RClick event, / COM version, on:
C#	private void RClick(object sender, EventArgs e) { }
C++	void OnRClick() { }
C++ Builder	voidfastcall 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

V D O

\	
v	БА

Private Sub RClick() End Sub



Syntax f	or 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 RemoveEvent (Ev as Event)

Occurs once an event is removed.

Туре	Description
Ev as <u>Event</u>	An Event object to be removed.

The RemoveEvent event notifies your application once an event is about to be removed. You can use the RemoveEvent event to release any extra data associated with the Event. The <u>Remove</u> method removes the specified event. The <u>RemoveSelection</u> method removes or erases all selected events in the schedule view. The <u>Clear</u> method of the Events collection clears all events in the schedule component. The <u>ClearAll</u> method clear all objects in the control, including the events. Any of these methods invoke calling of the RemoveEvent event. Use the <u>GroupID</u> property of the Event object to move a (remove/add) an Event from a Group to another. The <u>AllowMoveEventToOtherGroup</u> property specifies the keys combination so the user can move events from a group to another. The <u>ChangeEvent(exRemoveEvent</u>) event is equivalent with the RemoveEvent event.

Syntax for RemoveEvent event, /NET version, on:

C#	private void RemoveEvent(object sender,exontrol.EXSCHEDULELib.Event Ev)
	{
	}

Private Sub RemoveEvent(ByVal sender As System.Object,ByVal Ev As exontrol.EXSCHEDULELib.Event) Handles RemoveEvent
 End Sub

Syntax for RemoveEvent event, /COM version, on:

C#	private void RemoveEvent(object sender,	
	AxEXSCHEDULELibIScheduleEvents_RemoveEventEvent e)	
	{	
	}	
C++	void OnRemoveEvent(LPDISPATCH Ev)	

{

}
procedure RemoveEvent(ASender: TObject; Ev : lEvent); begin end;
procedure RemoveEvent(sender: System.Object; e: AxEXSCHEDULELibIScheduleEvents_RemoveEventEvent); begin end;
begin event RemoveEvent(oleobject Ev) end event RemoveEvent
Private Sub RemoveEvent(ByVal sender As System.Object, ByVal e As AxEXSCHEDULELibIScheduleEvents_RemoveEventEvent) Handles RemoveEvent End Sub
Private Sub RemoveEvent(ByVal Ev As EXSCHEDULELibCtl.IEvent) End Sub
Private Sub RemoveEvent(ByVal Ev As Object) End Sub
LPARAMETERS Ev
PROCEDURE OnRemoveEvent(oSchedule,Ev) RETURN

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

Java...

.

<SCRIPT EVENT="RemoveEvent(Ev)" LANGUAGE="JScript"> </SCRIPT>

VBSc	Sc <script <="" language="VBScript" th=""></script>	
------	---	--

Visual Data…	Procedure OnComRemoveEvent Variant IIEv Forward Send OnComRemoveEvent IIEv End_Procedure	
Visual Objects	METHOD OCX_RemoveEvent(Ev) CLASS MainDialog RETURN NIL	
X++	void onEvent_RemoveEvent(COM _Ev) { }	
XBasic	function RemoveEvent as v (Ev as OLE::Exontrol.Schedule.1::IEvent) end function	
dBASE	function nativeObject_RemoveEvent(Ev) return	

L

event ScrollButtonClick (ScrollBar as ScrollBarEnum, ScrollPart as ScrollPartEnum)

Occurs when the user clicks a button in the scrollbar.

Тур)e	Description
Scr	ollBar as <u>ScrollBarEnum</u>	A ScrollBarEnum expression that specifies the scroll bar being clicked.
Scr	ollPart as <u>ScrollPartEnum</u>	A ScrollPartEnum expression that indicates the part of the scroll being clicked.

Use the ScrollButtonClick event to notify your application that the user clicks a button in the control's scrollbar. The ScrollButtonClick event is fired when the user clicks and releases the mouse over an enabled part of the scroll bar. Use the <u>ScrollBars</u> property to specify the visible scrollbars in the control. Use the <u>ScrollPartVisible</u> property to add or remove buttons/parts in the control's scrollbar. Use the <u>ScrollPartEnable</u> property to specify enable or disable parts in the control's scrollbar. Use the <u>ScrollPartCaption</u> property to specify the caption of the scroll's part. Use the <u>Background</u> property to change the visual appearance for any part in the control's scroll bar.

Syntax for ScrollButtonClick event, /NET version, on:

private void ScrollButtonClick(object sender,exontrol.EXSCHEDULELib.ScrollBarEnum ScrollBar,exontrol.EXSCHEDULELib.ScrollPartEnum ScrollPart) {

VB

C#

C++

C#

}

Private Sub ScrollButtonClick(ByVal sender As System.Object,ByVal ScrollBar As exontrol.EXSCHEDULELib.ScrollBarEnum,ByVal ScrollPart As exontrol.EXSCHEDULELib.ScrollPartEnum) Handles ScrollButtonClick End Sub

Syntax for ScrollButtonClick event, /COM version, on:

private void ScrollButtonClick(object sender, AxEXSCHEDULELib._IScheduleEvents_ScrollButtonClickEvent e) { }

	{ }
C++ Builder	<pre>voidfastcall ScrollButtonClick(TObject *Sender,Exschedulelib_tlb::ScrollBarEnum ScrollBar,Exschedulelib_tlb::ScrollPartEnum ScrollPart) { }</pre>
Delphi	procedure ScrollButtonClick(ASender: TObject; ScrollBar : ScrollBarEnum;ScrollPart : ScrollPartEnum); begin end;
Delphi 8 (.NET only)	procedure ScrollButtonClick(sender: System.Object; e: AxEXSCHEDULELibIScheduleEvents_ScrollButtonClickEvent); begin end;
Powe	begin event ScrollButtonClick(long ScrollBar,long ScrollPart) end event ScrollButtonClick
VB.NET	Private Sub ScrollButtonClick(ByVal sender As System.Object, ByVal e As AxEXSCHEDULELibIScheduleEvents_ScrollButtonClickEvent) Handles ScrollButtonClick End Sub
VB6	Private Sub ScrollButtonClick(ByVal ScrollBar As EXSCHEDULELibCtl.ScrollBarEnum,ByVal ScrollPart As EXSCHEDULELibCtl.ScrollPartEnum) End Sub
VBA	Private Sub ScrollButtonClick(ByVal ScrollBar As Long,ByVal ScrollPart As Long) End Sub
VFP	LPARAMETERS ScrollBar,ScrollPart

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

Java	<script event="ScrollButtonClick(ScrollBar,ScrollPart)" language="JScript"> </script>
VBSc	<script language="VBScript"> Function ScrollButtonClick(ScrollBar,ScrollPart) End Function </script>
Visual Data	Procedure OnComScrollButtonClick OLEScrollBarEnum IIScrollBar OLEScrollPartEnum IIScrollPart Forward Send OnComScrollButtonClick IIScrollBar IIScrollPart End_Procedure
Visual Objects	METHOD OCX_ScrollButtonClick(ScrollBar,ScrollPart) CLASS MainDialog RETURN NIL
X++	void onEvent_ScrollButtonClick(int _ScrollBar,int _ScrollPart) { }
XBasic	function ScrollButtonClick as v (ScrollBar as OLE::Exontrol.Schedule.1::ScrollBarEnum,ScrollPart as OLE::Exontrol.Schedule.1::ScrollPartEnum) end function
dBASE	function nativeObject_ScrollButtonClick(ScrollBar,ScrollPart) return

The following VB sample displays the identifier of the scroll's button being clicked:

With Schedule1 .BeginUpdate .ScrollBars = exDisableBoth

```
.ScrollPartVisible(exVScroll, exLeftB1Part Or exRightB1Part) = True
.ScrollPartCaption(exVScroll, exLeftB1Part) = "<img></img>1"
.ScrollPartCaption(exVScroll, exRightB1Part) = "<img></img>2"
.EndUpdate
End With
```

Private Sub Schedule1_ScrollButtonClick(ByVal ScrollPart As EXSCHEDULELibCtl.ScrollPartEnum) MsgBox (ScrollPart) End Sub

The following VB.NET sample displays the identifier of the scroll's button being clicked:

With AxSchedule1
.BeginUpdate()
.ScrollBars = EXSCHEDULELib.ScrollBarsEnum.exDisableBoth
.set_ScrollPartVisible(EXSCHEDULELib.ScrollBarEnum.exVScroll,
EXSCHEDULELib.ScrollPartEnum.exLeftB1Part Or
EXSCHEDULELib.ScrollPartEnum.exRightB1Part, True)
.set_ScrollPartCaption(EXSCHEDULELib.ScrollBarEnum.exVScroll,
EXSCHEDULELib.ScrollPartEnum.exLeftB1Part, "1")
.set_ScrollPartCaption(EXSCHEDULELib.ScrollBarEnum.exVScroll,
EXSCHEDULELib.ScrollPartEnum.exRightB1Part, "2")
.EndUpdate()
End With

Private Sub AxSchedule1_ScrollButtonClick(ByVal sender As System.Object, ByVal e As AxEXSCHEDULELib._IScheduleEvents_ScrollButtonClickEvent) Handles AxSchedule1.ScrollButtonClick MessageBox.Show(e.scrollPart.ToString()) End Sub

The following C# sample displays the identifier of the scroll's button being clicked:

axSchedule1.BeginUpdate(); axSchedule1.ScrollBars = EXSCHEDULELib.ScrollBarsEnum.exDisableBoth; axSchedule1.set_ScrollPartVisible(EXSCHEDULELib.ScrollBarEnum.exVScroll, EXSCHEDULELib.ScrollPartEnum.exLeftB1Part | EXSCHEDULELib.ScrollPartEnum.exRightB1Part, true); axSchedule1.set_ScrollPartCaption(EXSCHEDULELib.ScrollBarEnum.exVScroll, EXSCHEDULELib.ScrollPartEnum.exLeftB1Part, "1"); axSchedule1.set_ScrollPartCaption(EXSCHEDULELib.ScrollBarEnum.exVScroll, EXSCHEDULELib.ScrollPartEnum.exRightB1Part, "2"); axSchedule1.EndUpdate();

private void axSchedule1_ScrollButtonClick(object sender, AxEXSCHEDULELib._IScheduleEvents_ScrollButtonClickEvent e)

```
MessageBox.Show(e.scrollPart.ToString());
```

The following C++ sample displays the identifier of the scroll's button being clicked:

```
m_schedule.BeginUpdate();
m_schedule.SetScrollBars( 15 /*exDisableBoth*/ );
m_schedule.SetScrollPartVisible( 0 /*exVScroll*/, 32768 /*exLeftB1Part*/ | 32
/*exRightB1Part*/, TRUE );
m_schedule.SetScrollPartCaption( 0 /*exVScroll*/, 32768 /*exLeftB1Part*/ , _T("<img>
</img>1") );
m_schedule.SetScrollPartCaption( 0 /*exVScroll*/, 32 /*exRightB1Part*/ , _T("<img>
</img>2") );
m_schedule.EndUpdate();
```

```
void OnScrollButtonClickSchedule1(long ScrollPart)
```

```
{
```

}

{

```
CString strFormat;
strFormat.Format( _T("%i"), ScrollPart );
MessageBox( strFormat );
```

The following VFP sample displays the identifier of the scroll's button being clicked:

```
With thisform.Schedule1

.BeginUpdate

.ScrollBars = 15

.ScrollPartVisible(0, bitor(32768,32)) = .t.
```

```
.ScrollPartCaption(0,32768) = "<img></img>1"
.ScrollPartCaption(0, 32) = "<img></img>2"
.EndUpdate
EndWith
```

*** ActiveX Control Event *** LPARAMETERS scrollpart

wait window nowait ltrim(str(scrollpart))

event UpdateEvent (Ev as Event)

Notifies your application once the event changes the starting or ending margins.

Туре	Description
Ev as <u>Event</u>	An Event object being updated.

The UpdateEvent event occurs once the event's margins are updated. The <u>Start</u> and <u>End</u> properties of the Event indicates the margins of the event. The UpdateEvent event may occur if the user moves or resizes the event. The UpdateEvent event occurs right after adding a new event (<u>AddEvent</u>). Removing an event does not fire the UpdateEvent event. The <u>Movable</u> property of the Event indicates whether the user can move the event at runtime. The <u>Resizable</u> property of the Event indicates whether the user can resize the event at runtime (start, end or both). You can use the UpdateEvent to update a dirty flag that indicates that the control's content must be saved.

The <u>AllowMoveEvent</u> property specifies the keys combination to let user moves the events at runtime. The <u>AllowResizeEvent</u> property specifies the keys combination to let user resizes the events at runtime. The <u>AllowMoveEventToOtherGroup</u> property specifies the keys combination so the user can move events from a group to another.

You can use the <u>UpdateEventsLabel</u> property to specify the label to be shown when the events are moved or resized at runtime. You can use the <u>Background</u>(exScheduleUpdateEventsBackColor) and <u>Background</u>(exScheduleUpdateEventsForeColor) properties to specify the visual appearance of the events being updated at runtime * moving or resizing).

In case you want to ensure that the user updates an event at runtime, you need to use the LayoutStartChanging and LayoutEndChanging events. The

LayoutStartChanging(exScheduleMoveEvent) event occurs once the user starts moving an event by dragging the event to a new position. The

LayoutEndChanging(exScheduleMoveEvent) event occurs once the user moved the event by dragging the event to a new position. The

LayoutStartChanging(exScheduleResizeStartEvent) event occurs once the user starts resizing the starting point of the event by dragging. The

LayoutEndChanging(exScheduleResizeStartEvent) event occurs once the user resized the starting point of the event by dragging. The

LayoutStartChanging(exScheduleResizeEndEvent) event occurs once the user starts resizing the ending margin of the event by dragging. The

LayoutEndChanging(exScheduleResizeEndEvent) event occurs once the user resized the ending margin of the event by dragging.

The order of the events when the user moves the event using the UI is:

- LayoutStartChanging(exScheduleMoveEvent)
- UpdateEvent
- LayoutEndChanging(exScheduleMoveEvent)

The order of the events when the user resizes the event using the UI is:

- LayoutStartChanging(exScheduleResizeStartEvent or exScheduleResizeEndEvent)
- UpdateEvent

VB

C++

• LayoutEndChanging(exScheduleResizeStartEvent or exScheduleResizeEndEvent)

Syntax for UpdateEvent event, /NET version, on:

C# private void UpdateEvent(object sender,exontrol.EXSCHEDULELib.Event Ev)
{
}

Private Sub UpdateEvent(ByVal sender As System.Object,ByVal Ev As exontrol.EXSCHEDULELib.Event) Handles UpdateEvent End Sub

Syntax for UpdateEvent event, /COM version, on:

void OnUpdateEvent(LPDISPATCH Ev)

C#	private void UpdateEvent(object sender,
	AxEXSCHEDULELibIScheduleEvents_UpdateEventEvent e)
	{
	}

	{ }
C++ Builder	voidfastcall UpdateEvent(TObject *Sender,Exschedulelib_tlb::IEvent *Ev) { }

Delphi	procedure UpdateEvent(ASender: TObject; Ev : IEvent);
	begin
	end;



Data...

procee AxEXS begin end;	dure UpdateEvent(sender: System.Object; e: CHEDULELibIScheduleEvents_UpdateEventEvent);
Powe	begin event UpdateEvent(oleobject Ev) end event UpdateEvent
VB.NET	Private Sub UpdateEvent(ByVal sender As System.Object, ByVal e As AxEXSCHEDULELibIScheduleEvents_UpdateEventEvent) Handles UpdateEvent End Sub
VB6	Private Sub UpdateEvent(ByVal Ev As EXSCHEDULELibCtl.IEvent) End Sub
VBA	Private Sub UpdateEvent(ByVal Ev As Object) End Sub
VFP	LPARAMETERS Ev
Xbas	PROCEDURE OnUpdateEvent(oSchedule,Ev) RETURN
Syntax	for UpdateEvent event, / COM version ^(others) , on:
Java	<script event="UpdateEvent(Ev)" language="JScript"> </script>
VBSc	<script language="VBScript"> Function UpdateEvent(Ev) End Function </script>
Visual	Procedure OnComUpdateEvent Variant IIEv

Forward Send OnComUpdateEvent IIEv
End_Procedure

Visual Objects	METHOD OCX_UpdateEvent(Ev) CLASS MainDialog RETURN NIL
X++	void onEvent_UpdateEvent(COM _Ev) { }
XBasic	function UpdateEvent as v (Ev as OLE::Exontrol.Schedule.1::IEvent) end function
dBASE	function nativeObject_UpdateEvent(Ev) return