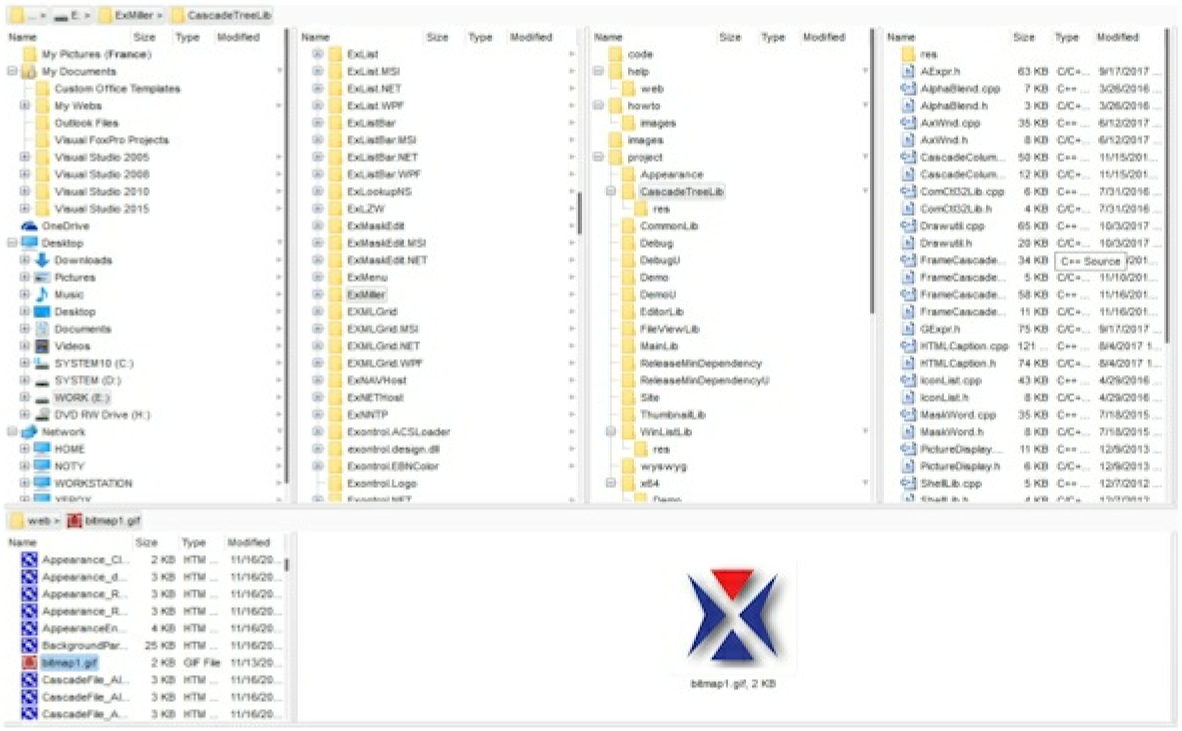
ExMiller

The eXMiller component is file-folder-view component that uses miller columns visualization to display system folders and files. The Miller columns (also known as Cascading Lists) are a browsing/visualization technique that can be applied to tree structures. The columns allow multiple levels of the hierarchy to be open at once, and provide a visual representation of the current location. It is closely related to techniques used earlier in the Smalltalk browser, but was independently invented by Mark S. Miller in 1980 at Yale University. Miller columns are most well known today as the "Columns view" mode of the Mac OS X Finder, as well as the "Browser" view in iTunes.

Features include:

- Single or Multiple Cascade Mode support
- List or Tree Multiple Columns support
- Single or Multiple Split View support
- Thumbnail Preview Mode support
- Single or Multiple Selection support
- Incremental Search support
- Filter Prompt support
- Customizable Context Menu support
- StatusBar support
- ScrollBar Extension support



How to get support?

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

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

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

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

Regards,
Exontrol Development Team

https://www.exontrol.com

constants AlignmentEnum

Specifies the object's alignment.

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

constants AllowSplitViewEnum

The AllowSplitViewEnum type specifies how many vertically split-panels the control support. The AllowSplitView property specifies whether the user can split the control into multiple-views. The AllowSplitViewEnum type supports the following values:

Name	Value	Description
exNoSplitView	0	No vertically split-view is allowed.
exAllowOneSplitView	1	One additional vertically split-panel is allowed.
exAllowTwoSplitView	2	Two additional vertically split-panel are allowed.
exSplitDisableEnsureVisiblePath	256	Indicates that path of the view being resized always fits the control's client area. For instance, once the user resizes the view, you want the selected files to be always visible. The exSplitDisableEnsureVisiblePath flag can be combined with any other values.

constants AppearanceEnum

The AppearanceEnum enumeration is used to specify the appearance of the control's header bar.

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

constants BackgroundPartEnum

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

Name	Value	Description
exSplitBar	18	Specifies the visual appearance for control's split bar.
exToolTipAppearance	64	Specifies the visual appearance of the borders of the tooltips.
exToolTipBackColor	65	Specifies the tooltip's background color.
exToolTipForeColor	66	Specifies the tooltip's foreground color.
exCheckBoxState0	70	exCheckBoxState0. Specifies the visual appearance for the check box in 0 state (unchecked).
exCheckBoxState1	71	exCheckBoxState1. Specifies the visual appearance for the check box in 1 state (checked).
exCheckBoxState2	72	exCheckBoxState2. Specifies the visual appearance for the check box in 2 state (partial, not used).
exHSplitBar	141	Specifies the visual appearance for horizontal split bar.
exCSplitBar	142	Specifies the solid color / visual appearance of the split bar that creates new views.
exStatusBackColor	168	Specifies the status bar's background color. The StatusBarVisible property specifies whether the control's status bar is visible or hidden. The StatusBarLabel property specifies the HTML label the control's status bar is displaying.
exStatusForeColor	169	Specifies the status bar's foreground color. The StatusBarVisible property specifies whether the control's status bar is visible or hidden. The StatusBarLabel property specifies the HTML label the control's status bar is displaying.
exSplitBarSize	170	Specifies the size of the control's split bar, when

		resizing the cascade columns is enabled.
exDisableSplitBar	171	Specifies the visual appearance for control's split bar, when resizing the cascade columns is disabled.
exDisableSplitBarSize	172	Specifies the size of the control's split bar, when resizing the cascade columns is disabled.
exFocusFrame	173	Specifies the visual appearance of the frame around the focusing cascade column.
exStatusPanelBackColor	174	Specifies the status panel's background color. The StatusBarVisible property specifies whether the control's status bar is visible or hidden. The StatusBarLabel property specifies the HTML label the control's status bar is displaying.
exThumbnailBackColorAlt	175	Specifies the alternate background color for thumbnail's view.
exThumbnailForeColorAlt	176	Specifies the alternate foreground color for thumbnail's view.
exThumbnailBorderColor	177	Specifies the color to show the thumbnail's border.
exThumbnailSelBorderColor	178	Specifies the color to show the selected thumbnail.
exThumbnailSelBorderColorHide	179	Specifies the color to show the selected thumbnail, while the control has no focus.
exTreeGlyphOpen	180	Specifies the visual appearance for the +/- buttons when it is collapsed.
exTreeGlyphClose	181	Specifies the visual appearance for the +/- buttons when it is expanded.
exColumnsPositionSign	182	exColumnsPositionSign. Specifies the visual appearance for the position sign between columns, when the user changes the position of the column by drag an drop.
exTreeLinesColor	186	exTreeLinesColor. Specifies the color to show the tree-lines (connecting lines from the parent to the children)
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	The background part (exLowerBackPart and 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. The 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 CascadeModeEnum

The CascadeModeEnum type specifies the modes the control supports. The Mode property indicates the mode the control displays the cascade columns. The CascadeModeEnum type supports the following values:

Name	Value	Description
exFixCascadeMode	0	Each cascade column can be displayed with a different width. The DefColumnWidth property specifies the width to create a new cascade column.
exSingleCascadeMode	1	No cascade columns support.
exSplitEqualCascadeMode	2	The cascade column fits equally the control's client area. The FitCascadeColumns property retrieves or sets a value that indicates the number of cascading columns to fit.
exSplitFixCascadeMode	3	The cascade column fits equally the control's client area. The FitCascadeColumns property retrieves or sets a value that indicates the number of cascading columns to fit.
exDisableResizeCascadeColumns	256	The user can't resize the cascade columns.
exAutoFitOnResizeClient	512	Each cascade column gets resized as soon as the control gets resized.
exDisableThumbnails	1024	No thumbnail mode is allowed. The SelectMode property on exSelectModeThumbnail has no effect.

constants CheckBoxEnum

The CheckBoxEnum expression defines the type of check boxes that control supports. Use the HasCheckBox property to assign a check box for each item in your control.

Name	Value	Description
NoCheckBox	0	The control provides no check boxes.
CheckBox	-1	The control provides a two-states check box for each item.
PartialCheckBox	1	The control provides a partial check box (three-states check box) for each item.

constants FileColumnEnum

The FileColumnEnum type specifies the columns into the file-view control. You can use the ColumnsVisible property to show/hide multiple columns at once. The FileColumnEnum type supports the following flags:

Name	Value	Description
exFileColumnName	2	Indicates the Name column.
exFileColumnSize	4	Indicates the Size column.
exFileColumnType	8	Indicates the Type column.
exFileColumnModified	16	Indicates the Modified column.

constants PictureDisplayEnum

Specifies how the picture is displayed on the control's background. Use the PictureDisplay property to specify how the control displays its picture.

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

constants ScrollBarEnum

The ScrollBarEnum type specifies the vertical or horizontal scroll bar in the control. Use the ScrollPartVisible property to specify the visible parts in the control's scroll bars

Name	Value	Description
exVScroll	0	Indicates the vertical scroll bar (cascade column view)
exHScroll	1	Indicates the horizontal scroll bar (cascade column view)
exScroll	2	Indicates the control's horizontal scroll bar.

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.

Name	Value	Description
exExtentThumbPart	65536	The thumb-extension part.
exLeftB1Part	32768	(L1) The first additional button, in the left or top area. By default, this button is hidden.
exLeftB2Part	16384	(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 SelectModeEnum

The SelectModeEnum type specifies the mode the control displays the last visible cascade column. The SelectMode property indicates the mode the control displays the last visible cascade column. The SelectModeEnum type supports the following values:

Name	Value	Description
exSelectModeList	1	The last cascade column view displays a list with all visible columns.
exSelectModeThumbnail	2	The last cascade column view displays thumbnails for previously selected files. The exDisableThumbnails flag of Mode property prevents any thumbnail mode.

constants StatusBarAnchorEnum

The StatusBarAnchorEnum type specifies how the status bar is displayed relative to the control. The StatusBarVisible property specifies whether the control's status bar is visible or hidden. The StatusBarLabel property specifies the HTML label the control's status bar is displaying. The StatusBarAnchorEnum type supports the following values:

Name	Value	Description
exStatusBarNone	0	The control's status bar is not visible.
exStatusBarAnchorBottom	1	The control's status bar is aligned to the bottom side of the control.
exStatusBarAnchorTop	2	The control's status bar is aligned to the top side of the control.

constants TypeEnum

Specifies the type of items(files) the Get method returns.

Name	Value	Description
SelItems	0	Gets the collection of selected items.
AllItems	1	Gets the entire collection of items
CheckItems	2	Gets the checked items.
VisibleItems	3	Gets the visible items as they are listed.

constants UIVisualThemeEnum

The UIVisualThemeEnum expression specifies the UI parts that the control can shown using the current visual theme. The The UIVisualThemeEnum type supports following values:

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

constants ViewOperationEnum

The ViewOperationEnum type specifies operations that could start or end. The ViewStartChanging / ViewEndChanging events notify your application that an operation starts or ends. The ViewOperationEnum type supports the following values:

Name	Value	Description
exSplitViewChange	1	The user splits/resizes the view into multiple views.
exResizeCascadeColumn	2	The user resizes the cascade column.
exSelectItemChange	3	The control's selection is changing.
exSelectThumbnailChange	4	The selects a thumbnail file.
exCheckStateChange	15	The uses checks or un-checks the file.
exShowContextMenu	20	Occurs when the control is about to display the object's context menu.
exExecuteContextMenu	21	Occurs when the control is about to execute a command from the object's context menu.

Appearance object

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

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

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

Adds or replaces a skin object to the control.

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

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

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

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

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

The identifier you choose for the skin is very important to be used in the background properties like explained bellow. Shortly, the color properties uses 4 bytes (DWORD, double WORD, and so on) to hold a RGB value. More than that, the first byte (most significant byte in the color) is used only to specify system color. if the first bit in the byte is 1, the rest of bits indicates the index of the system color being used. So, we use the last 7 bits in the high significant byte of the color to indicates the identifier of the skin being used. So, since the 7 bits can cover 127 values, excluding 0, we have 126 possibilities to store an identifier in that byte. This way, a DWORD expression indicates the background color stored in RRGGBB format and the index of the skin (ID parameter) in the last 7 bits in the high significant byte of the color. For instance, the BackColor = BackColor Or &H2000000 indicates that we apply the skin with the index 2 using the old color, to the object that BackColor is applied.

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

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

		RBS_CHECKEDDIS
	BP_USERBUTTON = 5	
CLOCK	CLP_TIME = 1	CLS_NORMAL = 1
COMBOBOX	CP_DROPDOWNBUTTON = 1	CBXS_NORMAL = CBXS_HOT = 2 CBXS_PRESSED = CBXS_DISABLED =
EDIT	EP_CARET = 2	
	EP_EDITTEXT = 1	ETS_NORMAL = 1 2 ETS_SELECTED ETS_DISABLED = 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_SELECTE EBHP_SELECTED 6
	EBP_IEBARMENU = 4	EBM_NORMAL = 1 = 2 EBM_PRESSE
	EBP_NORMALGROUPBACKGROUND = 5	
	EBP_NORMALGROUPCOLLAPSE = 6	EBNGC_NORMAL EBNGC_HOT = 2 EBNGC_PRESSED
	EBP_NORMALGROUPEXPAND = 7	EBNGE_NORMAL EBNGE_HOT = 2 EBNGE_PRESSED
	EBP_NORMALGROUPHEAD = 8	
	EBP_SPECIALGROUPBACKGROUND = 9	
	EBP_SPECIALGROUPCOLLAPSE = 10	EBSGC_NORMAL EBSGC_HOT = 2 EBSGC_PRESSED
		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_PRESSE
	HP_HEADERITEMRIGHT = 3	HIRS_NORMAL = = 2 HIRS_PRESSE
	HP_HEADERSORTARROW = 4	HSAS_SORTEDUP HSAS_SORTEDD
LISTVIEW	LVP_EMPTYTEXT = 5	
	LVP_LISTDETAIL = 3	
	LVP_LISTGROUP = 2	
	LVP_LISTITEM = 1	LIS_NORMAL = 1 2 LIS_SELECTED = LIS_DISABLED = 4 LIS_SELECTEDNO 5
	LVP_LISTSORTEDDETAIL = 4	
MENU	MP_MENUBARDROPDOWN = 4	MS_NORMAL = 1 MS_SELECTED = MS_DEMOTED =
	MP_MENUBARITEM = 3	MS_NORMAL = 1 MS_SELECTED = MS_DEMOTED =
	MP_CHEVRON = 5	MS_NORMAL = 1 MS_SELECTED = MS_DEMOTED =
	MP_MENUDROPDOWN = 2	MS_NORMAL = 1 MS_SELECTED = MS_DEMOTED =
	MP_MENUITEM = 1	MS_NORMAL = 1 MS_SELECTED = MS_DEMOTED =
	MP_SEPARATOR = 6	MS_NORMAL = 1 MS_SELECTED = MS_DEMOTED =
		MDS_NORMAL = = 2 MDS_PRESSE

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

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

	SPNP_UP = 1	= 2 UPS_PRESSED UPS_DISABLED =
	SPNP_UPHORZ = 3	UPHZS_NORMAL = UPHZS_HOT = 2 UPHZS_PRESSED UPHZS_DISABLED
STARTPANEL	SPP_LOGOFF = 8	
	SPP_LOGOFFBUTTONS = 9	SPLS_NORMAL = SPLS_HOT = 2 SPLS_PRESSED =
	SPP_MOREPROGRAMS = 2	
	SPP_MOREPROGRAMSARROW = 3	SPS_NORMAL = 1 = 2 SPS_PRESSED
	SPP_PLACESLIST = 6	
	SPP_PLACESLISTSEPARATOR = 7	
	SPP_PREVIEW = 11	
	SPP_PROGLIST = 4	
	SPP_PROGLISTSEPARATOR = 5	
	SPP_USERPANE = 1	
	SPP_USERPICTURE = 10	
STATUS	SP_GRIPPER = 3	
	SP_PANE = 1	
	SP_GRIPPERPANE = 2	
TAB	TABP_BODY = 10	
	TABP_PANE = 9	
	TABP_TABITEM = 1	TIS_NORMAL = 1 2 TIS_SELECTED TIS_DISABLED = 4 TIS_FOCUSED = 5
	TABP_TABITEMBOTHEDGE = 4	TIBES_NORMAL = TIBES_HOT = 2 TIBES_SELECTED TIBES_DISABLED TIBES_FOCUSED
	TABP_TABITEMLEFTEDGE = 2	TILES_NORMAL = TILES_HOT = 2 TILES_SELECTED TILES_DISABLED TILES_FOCUSED

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

	TP_DROPDOWNBUTTON = 2	TS_NORMAL = 1 T
		TS_PRESSED = 3
		TS_DISABLED = 4
		TS_CHECKED = 5
		TS_HOTCHECKED

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

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

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

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

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

	TKP_THUMBBOTTOM = 4	TUBS_HOT = 2 TUBS_PRESSED = TUBS_FOCUSED = TUBS_DISABLED =
	TKP_THUMBLEFT = 7	TUVLS_NORMAL = TUVLS_HOT = 2 TUVLS_PRESSED TUVLS_FOCUSED TUVLS_DISABLED
	TKP_THUMBRIGHT = 8	TUVRS_NORMAL = TUVRS_HOT = 2 TUVRS_PRESSED TUVRS_FOCUSED TUVRS_DISABLED
	TKP_THUMBTOP = 5	TUTS_NORMAL = TUTS_HOT = 2 TUTS_PRESSED = TUTS_FOCUSED = TUTS_DISABLED =
	TKP_THUMBVERT = 6	TUVS_NORMAL = TUVS_HOT = 2 TUVS_PRESSED = TUVS_FOCUSED = TUVS_DISABLED =
	TKP_TICS = 9	TSS_NORMAL = 1
	TKP_TICSVERT = 10	TSVS_NORMAL =
	TKP_TRACK = 1	TRS_NORMAL = 1
	TKP_TRACKVERT = 2	TRVS_NORMAL =
TRAYNOTIFY	TNP_ANIMBACKGROUND = 2	
	TNP_BACKGROUND = 1	
TREEVIEW	TVP_BRANCH = 3	
	TVP_GLYPH = 2	GLPS_CLOSED = GLPS_OPENED =
	TVP_TREEITEM = 1	TREIS_NORMAL = TREIS_HOT = 2 TREIS_SELECTED TREIS_DISABLED TREIS_SELECTED = 5
		CS_ACTIVE = 1 CS

WINDOW	WP_CAPTION = 1	= 2 CS_DISABLED
	WP_CAPTIONSIZINGTEMPLATE = 30	
	WP_CLOSEBUTTON = 18	CBS_NORMAL = 1 = 2 CBS_PUSHED CBS_DISABLED =
	WP_DIALOG = 29	
	WP_FRAMEBOTTOM = 9	FS_ACTIVE = 1 FS = 2
	WP_FRAMEBOTTOMSIZINGTEMPLATE = 36	
	WP_FRAMELEFT = 7	FS_ACTIVE = 1 FS = 2
	WP_FRAMELEFTSIZINGTEMPLATE = 32	
	WP_FRAMERIGHT = 8	FS_ACTIVE = 1 FS = 2
	WP_FRAMERIGHTSIZINGTEMPLATE = 34	
	WP_HELPBUTTON = 23	HBS_NORMAL = 1 = 2 HBS_PUSHED HBS_DISABLED =
	WP_HORZSCROLL = 25	HSS_NORMAL = 1 = 2 HSS_PUSHED HSS_DISABLED =
	WP_HORZTHUMB = 26	HTS_NORMAL = 1 2 HTS_PUSHED = HTS_DISABLED =
	WP_MAX_BUTTON	MAXBS_NORMAL MAXBS_HOT = 2 MAXBS_PUSHED MAXBS_DISABLED
	WP_MAXCAPTION = 5	MXCS_ACTIVE = 1 MXCS_INACTIVE MXCS_DISABLED
	WP_MDICLOSEBUTTON = 20	CBS_NORMAL = 1 = 2 CBS_PUSHED CBS_DISABLED =
	WP_MDIHELPBUTTON = 24	HBS_NORMAL = 1 = 2 HBS_PUSHED HBS_DISABLED =
	WP_MDIMINBUTTON = 16	MINBS_NORMAL = MINBS_HOT = 2

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

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

method Appearance.Clear ()

Removes all skins in the control.

Type	Description

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

method Appearance.Remove (ID as Long)

Removes a specific skin from the control.

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

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

property Appearance.RenderType as Long

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

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

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

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

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

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

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

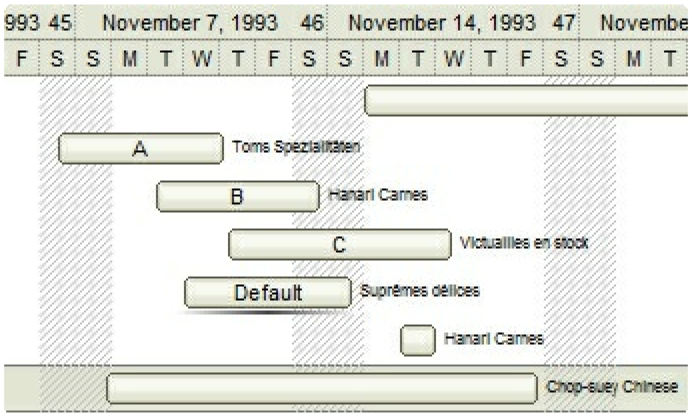
In the following screen shot the following objects displays the current EBN with a different color:

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

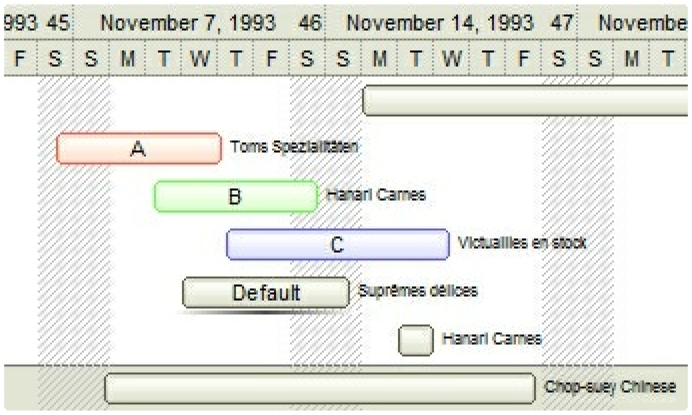
- "C" in Blue (RGB(0,0,255), for instance the bar's property exBarColor is 0x1FF0000
- "Default", no color is specified, for instance the bar's property exBarColor is 0x1000000

The RenderType property could be one of the following:

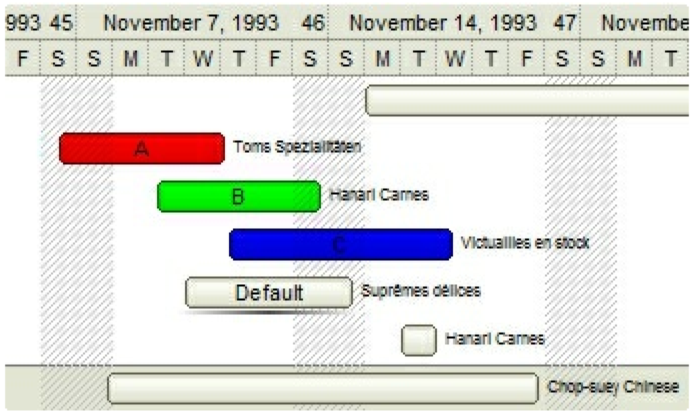
- ***-3**, no color is applied*. For instance, the BackColorHeader = &H1FF0000 is displayed as would be .BackColorHeader = &H1000000, so the 0xFF0000 color (Blue color) is ignored. You can use this option to allow the control displays the EBN colors or not.



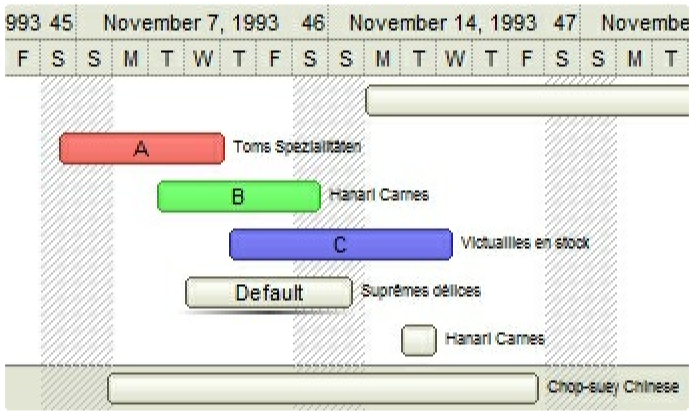
- ***-2**, OR-color scheme*. The color to be applied on the part of the control is a OR bit combination between the original EBN color and the specified color. For instance, the BackColorHeader = &H1FF0000, applies the OR bit for the entire Blue channel, or in other words, it applies a less Blue to the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



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

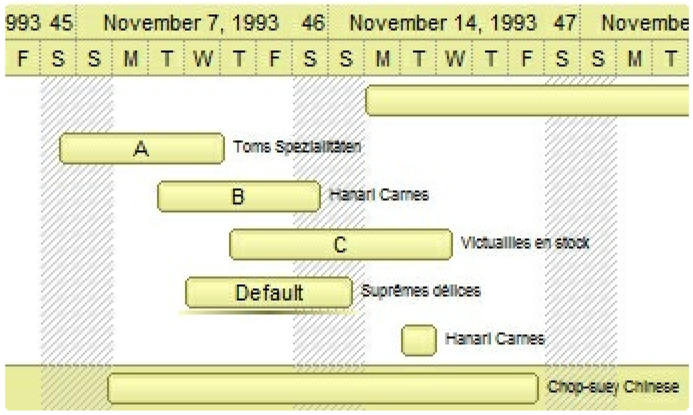


- ***0**, default*, the specified color is applied to the EBN. For instance, the BackColorHeader = &H1FF0000, applies a Blue color to the object. This option could be used to specify any color for the part of the components, that support EBN objects, not only solid colors.

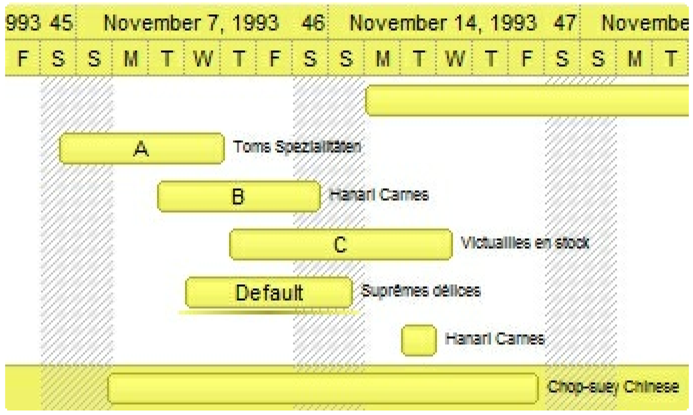


- **0xAABBGGRR**, where the AA a value between 0 to 255, which indicates the transparency, and RR, GG, BB the red, green and blue values. This option applies the same color to all parts that displays EBN objects, whit ignoring any specified color in the color property. For instance, the RenderType on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

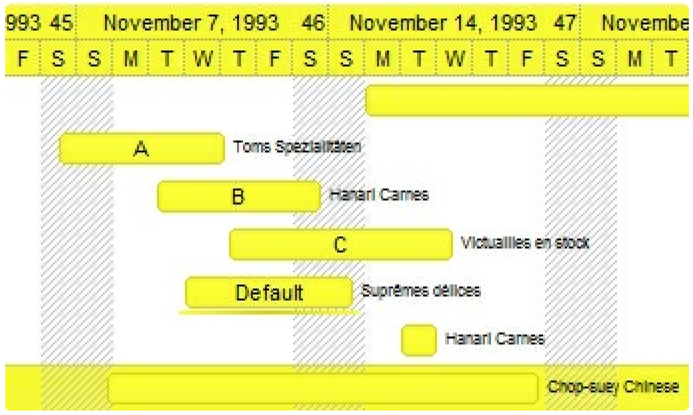
*The following picture shows the control with the RenderType property on 0x**40**00FFFF (25% Yellow, 0x40 or 64 in decimal is 25% from 256):*



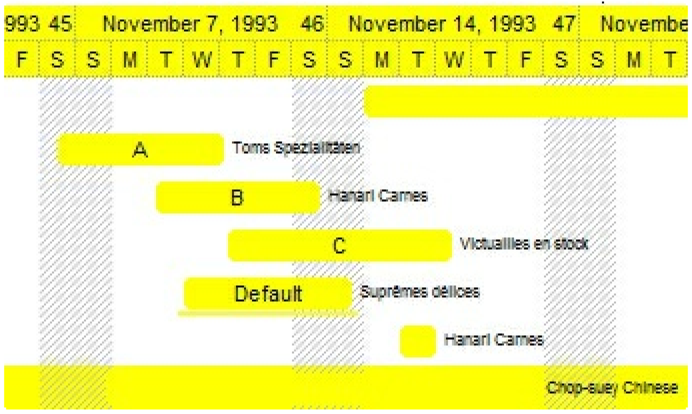
*The following picture shows the control with the RenderType property on 0x**80**00FFFF (50% Yellow, 0x80 or 128 in decimal is 50% from 256):*



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



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


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

CascadeFile 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: {A44EFDCD-D526-4E58-909D-193FD6519290}. The object's program identifier is: "Exontrol.ExMiller". The /COM object module is: "ExMiller.dll"

The Miller columns (also known as Cascading Lists) are a browsing/visualization technique that can be applied to tree structures. The columns allow multiple levels of the hierarchy to be open at once, and provide a visual representation of the current location. It is closely related to techniques used earlier in the Smalltalk browser, but was independently invented by Mark S. Miller in 1980 at Yale University. Miller columns are most well known today as the "Columns view" mode of the Mac OS X Finder, as well as the "Browser" view in iTunes. The CascadeFile object supports the following properties / methods:

Name	Description
AllowContextMenu	Enables or disables the file's context menu.
AllowSplitView	Specifies whether the user can split the control into multiple-views
AnchorFromPoint	Retrieves the identifier of the anchor from point.
Appearance	Retrieves or sets the control's appearance.
AttachTemplate	Attaches a script to the current object, including the events, from a string, file, a safe array of bytes.
AutoUpdate	Determines whether the control's content is automatically updated once a change occurs.
BackColor	Specifies the control's background color.
BackColorAlternate	Specifies the control's alternate background color.
BackColorHeader	Specifies the header's background color.
Background	Returns or sets a value that indicates the background color for parts in the control.
BeginUpdate	Maintains performance when items are added to the control one at a time. This method prevents the control from painting until the EndUpdate method is called.
BorderHeight	Sets or retrieves a value that indicates the border height of the control.
BorderWidth	Sets or retrieves a value that indicates the border width of the control.
ColumnAutoResize	Returns or sets a value indicating whether the control will automatically size its visible columns to fit on the control's client width.

ColumnsVisible	Indicates the columns being visible.
DefaultItemHeight	Retrieves or sets a value that indicates the default item height.
DefColumnWidth	Specifies the width to create a new cascade column.
Enabled	Enables or disables the control.
EndUpdate	Resumes painting the control after painting is suspended by the BeginUpdate method.
EventParam	Retrieves or sets a value that indicates the current's event parameter.
ExecuteContextCommand	Executes a context menu command.
ExecuteContextMenu	Executes a command from the object's context menu.
ExecuteTemplate	Executes a template and returns the result.
ExpandFolders	Retrieves or sets a value that indicates whether the control expands the folder objects.
ExploreFromHere	Specifies the root folder for the control.
FitCascadeColumns	Retrieves or sets a value that indicates the number of cascading columns to fit.
FitToClient	Resizes or/and moves the all cascade columns to fit the control's client area.
Font	Retrieves or sets the control's font.
ForeColor	Specifies the control's foreground color.
ForeColorAlternate	Specifies the control's alternate foreground color.
ForeColorHeader	Specifies the header's foreground color.
FormatABC	Formats the A,B,C values based on the giving expression and returns the result.
FormatAnchor	Specifies the visual effect for anchor elements in HTML captions.
FreezeEvents	Prevents the control to fire any event.
FullRowSelect	Enables full-row selection in the control.
Get	Gets the list of files (selected, checked, visible, ...) as a safe array of VARIANT.
HasCheckBox	Specifies whether the control displays a check box for each item.
	Retrieves or sets a value that indicates the header's

HeaderAppearance	appearance.
HeaderHeight	Retrieves or sets a value indicating the control's header height.
HeaderVisible	Retrieves or sets a value that indicates whether the control's header bar is visible or hidden.
HideSelection	Returns a value that determines whether selected item appears highlighted when a control loses the focus.
HotBackColor	Retrieves or sets a value that indicates the hot background color.
HotForeColor	Retrieves or sets a value that indicates the hot foreground color.
HTMLPicture	Adds or replaces a picture in HTML captions.
hWnd	Retrieves the control's window handle.
Images	Sets at runtime the control's image list. The Handle should be a handle to an Images List Control.
ImageSize	Retrieves or sets the size of icons the control displays.
Indent	Retrieves or sets the amount, in pixels, that child items are indented relative to their parent items.
Layout	Saves or loads the control's layout, such current selection for each panel, the widths of the cascade columns, and so on.
MaxColumnWidth	Specifies the maximum width for any cascade column.
MinColumnWidth	Specifies the minimum width for any cascade column.
Mode	Indicates the mode the control displays the cascade columns.
Picture	Retrieves or sets a graphic to be displayed in the control.
PictureDisplay	Retrieves or sets a value that indicates the way how the graphic is displayed on the control's background
Refresh	Refreshes the control.
ReplaceIcon	Adds a new icon, replaces an icon or clears the control's image list.
ScrollButtonHeight	Specifies the height of the button in the vertical scrollbar.
ScrollButtonWidth	Specifies the width of the button in the horizontal scrollbar.
ScrollFont	Retrieves or sets the scrollbar's font.

ScrollHeight	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.
ScrollPartEnable	Indicates whether the specified scroll part is enabled or disabled.
ScrollPartVisible	Indicates whether the specified scroll part is visible or hidden.
ScrollThumbSize	Specifies the size of the thumb in the scrollbar.
ScrollToolTip	Specifies the tooltip being shown when the user moves the scroll box.
ScrollWidth	Specifies the width of the vertical scrollbar.
SelBackColor	Retrieves or sets a value that indicates the selection background color.
Select	Specifies the current selection.
SelectMode	Indicates the mode the control displays the last visible cascade column.
SelForeColor	Retrieves or sets a value that indicates the selection foreground color.
ShowContextMenu	Specifies the object's context menu.
ShowFocusRect	Retrieves or sets a value indicating whether the control draws a thin rectangle arround the focused item.
ShowImageList	Specifies whether the control's image list window is visible or hidden.
ShowToolTip	Shows the specified tooltip at given position.
SingleSel	Retrieves or sets a value indicating whether control support single or multiples selection.
SplitViewHeight	Specifies the height of split panels, separated by comma.
StatusBarLabel	Specifies the HTML label the control's status bar is displaying.
StatusBarVisible	Specifies whether the control's status bar is visible or hidden.
Template	Specifies the control's template.
TemplateDef	Defines inside variables for the next Template/ExecuteTemplate call.

TemplatePut	Defines inside variables for the next Template/ExecuteTemplate call.
ToolTipDelay	Specifies the time in ms that passes before the ToolTip appears.
ToolTipFont	Retrieves or sets the tooltip's font.
ToolTipPopDelay	Specifies the period in ms of time the ToolTip remains visible if the mouse pointer is stationary within a control.
ToolTipWidth	Specifies a value that indicates the width of the tooltip window, in pixels.
UseTabKey	Retrieves or sets a value that specifies whether the Tab or SHIFT + Tab key navigates through the cascading columns.
Version	Retrieves the control's version.
VisualAppearance	Retrieves the control's appearance.

property CascadeFile.AllowContextMenu as Boolean

Enables or disables the file's context menu.

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

By default, the AllowContextMenu property is True. Use the AllowContextMenu to disable the control's context menu. The control's context menu is displayed when the user does a right click on the file or the folder. The system controls the items being inserted to the control's context menu. Use the ExecuteContextCommand method to execute a command from the file's context menu. The ShowContextMenu property indicates the items to be displayed on the object's context menu. The ShowContextMenu property can be used to disable, update, remove or add new items. The ExecuteContextMenu property specifies the identifier of the command to be executed (id option in the ShowContextMenu property).

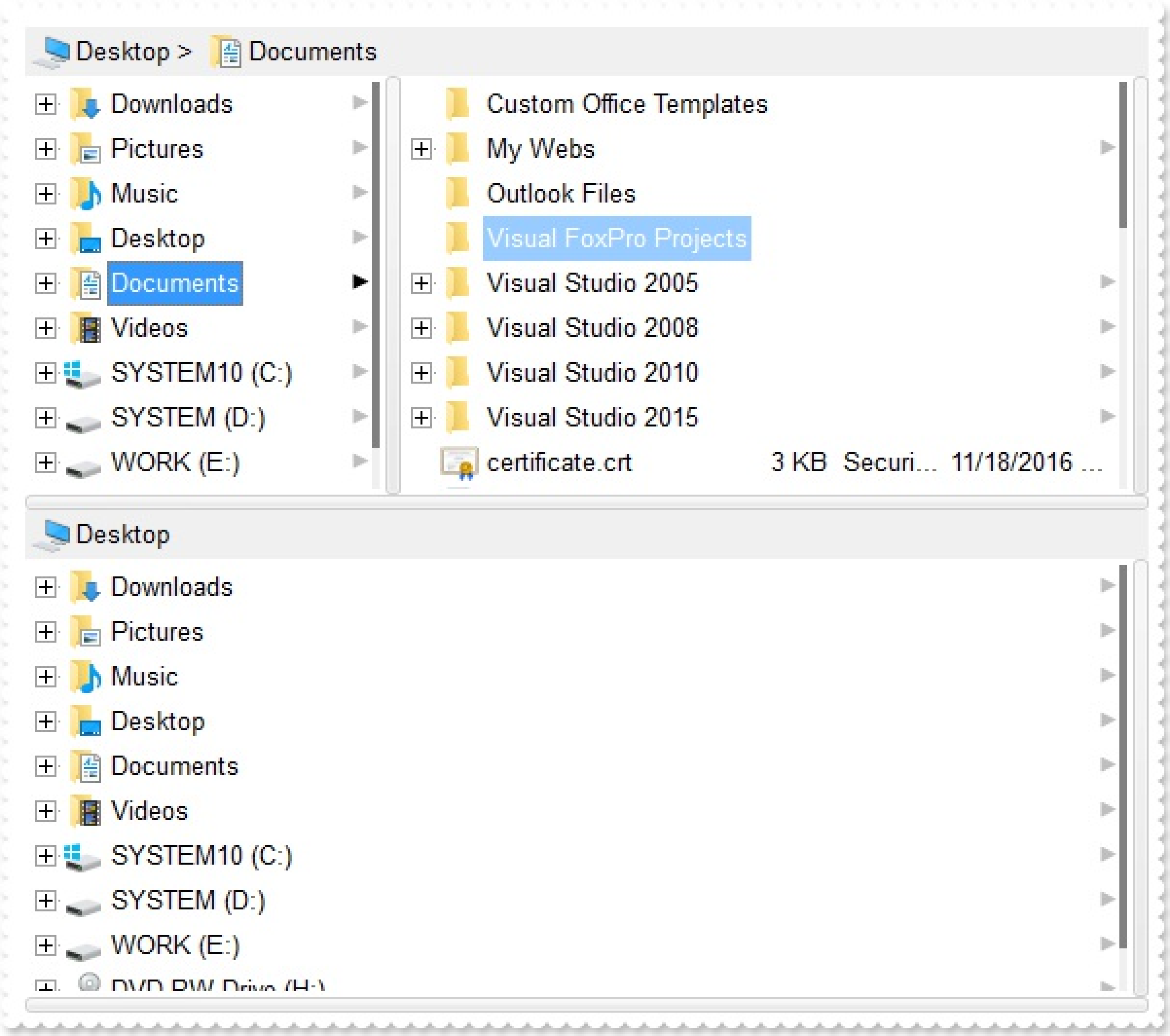
property CascadeFile.AllowSplitView as AllowSplitViewEnum

Specifies whether the user can split the control into multiple-views

Type	Description
AllowSplitViewEnum	An AllowSplitViewEnum expression whether the control supports multiple views (arranged vertically)

By default, the AllowSplitView property is exNoSplitView, so no additional view is supported. The AllowSplitView property specifies whether the user can split the control into multiple-views. The SplitViewHeight property specifies the height of split panels, separated by comma. The Background(exHSplitBar) property specifies the visual appearance of the control's split bar (horizontal split bar)

The following screen show shows the control with two vertically split-panels:



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

Retrieves the identifier of the anchor from point.

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

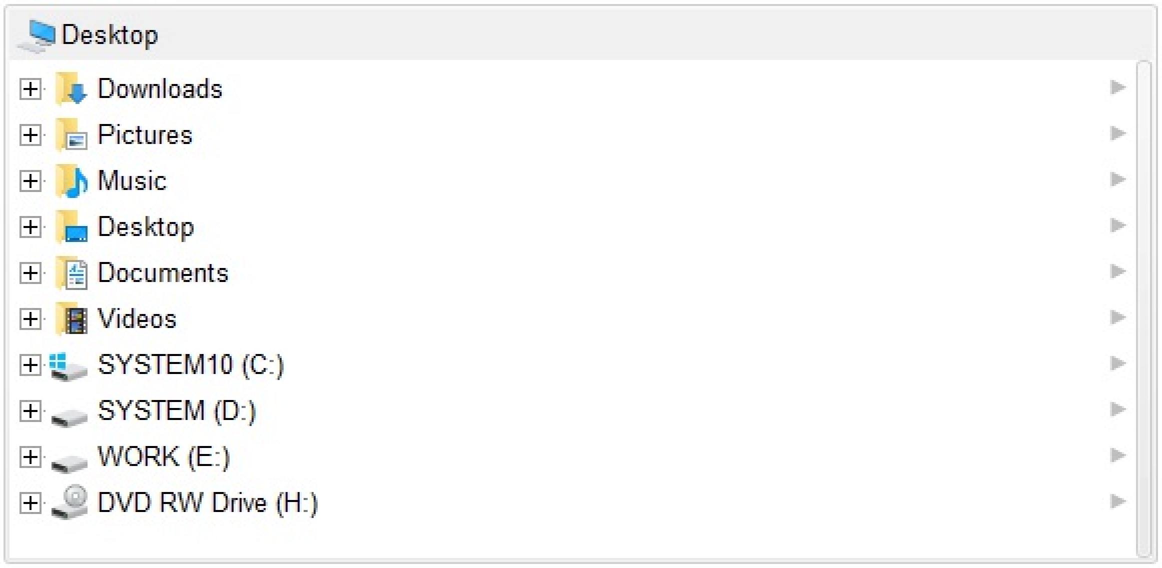
Use the AnchorFromPoint property to determine the identifier of the anchor from the point. Use the <a id;options> anchor elements to add hyperlinks to cell's caption. The control fires the AnchorClick event when the user clicks an anchor element. Use the ShowToolTip method to show the specified tooltip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

property CascadeFile.Appearance as AppearanceEnum

Retrieves or sets the control's appearance.

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

Use the Appearance property to specify the control's border. The Add method to add new skins to the control. Use the BackColor property to specify the control's background color. The BorderWidth / BorderHeight property specifies the size of the control's border.



method CascadeFile.AttachTemplate (Template as Variant)

Attaches a script to the current object, including the events, from a string, file, a safe array of bytes.

Type	Description
Template as Variant	A string expression that specifies the Template to execute.

The AttachTemplate/x-script code is a simple way of calling control/object's properties, methods/events using strings. The AttachTemplate features allows you to attach a x-script code to the component. The AttachTemplate method executes x-script code (including events), from a string, file or a safe array of bytes. This feature allows you to run any x-script code for any configuration of the component /COM, /NET or /WPF. Exontrol owns the x-script implementation in its easiest form and it does not require any VB engine or whatever to get executed. The x-script code can be converted to several programming languages using the eXHelper tool.

The following sample opens the Windows Internet Explorer once the user clicks the control (/COM version):

```
AttachTemplate("handle Click(){ CreateObject(`internetexplorer.application`){ Visible =
True; Navigate(`https://www.exontrol.com`) } } ")
```

This script is equivalent with the following VB code:

```
Private Sub CascadeFile1_Click()
    With CreateObject("internetexplorer.application")
        .Visible = True
        .Navigate ("https://www.exontrol.com")
    End With
End Sub
```

The AttachTemplate/x-script syntax in BNF notation is defined like follows:

```
<x-script> := <lines>
<lines> := <line>[<eol> <lines>] | <block>
<block> := <call> [<eol>] { [<eol>] <lines> [<eol>] } [<eol>]
<eol> := ";" | "\r\n"
<line> := <dim> | <createobject> | <call> | <set> | <comment> | <handle>[<eol>]{[<eol>]
<lines>[<eol>]}[<eol>]
<dim> := "DIM" <variables>
<variables> := <variable> [, <variables>]
```

```
<variable> := "ME" | <identifier>
<createobject> := "CREATEOBJECT(`"<type>"`)"
<call> := <variable> | <property> | <variable>"."<property> | <createobject>"."<property>
<property> := [<property>"."]<identifier>["("<parameters>")"]
<set> := <call> "=" <value>
<property> := <identifier> | <identifier>"("[<parameters>]")"
<parameters> := <value> [","<parameters>]
<value> := <boolean> | <number> | <color> | <date> | <string> | <createobject> | <call>
<boolean> := "TRUE" | "FALSE"
<number> := "0X"<hexa> | ["-"]<integer>["."<integer>]
<digit10> := 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
<digit16> := <digit10> | A | B | C | D | E | F
<integer> := <digit10>[<integer>]
<hexa> := <digit16>[<hexa>]
<color> := "RGB("<integer>","<integer>","<integer>")"
<date> := "#"<integer>"/"<integer>"/"<integer>" "[<integer>":"<integer>":"<integer>"]"#"
<string> := '"'<text>'"' | "`"<text>"`"
<comment> := "'"<text>
<handle> := "handle " <event>
<event> := <identifier>"("[<eparameters>]")"
<eparameters> := <eparameter> [","<eparameters>]
<parameters> := <identifier>
```

where:

<identifier> indicates an identifier of the variable, property, method or event, and should start with a letter.
<type> indicates the type the CreateObject function creates, as a progID for /COM version or the assembly-qualified name of the type to create for /NET or /WPF version
<text> any string of characters

The Template or x-script is composed by lines of instructions. Instructions are separated by "\n\r" (newline characters) or ";" character.

The advantage of the AttachTemplate relative to Template / ExecuteTemplate is that the AttachTemplate can add handlers to the control events.

property CascadeFile.AutoUpdate as Boolean

Determines whether the control is refreshed while a file or folder is changed, moved, or renamed.

Type	Description
Boolean	A boolean expression that determines whether the control is refreshed while a file or folder is changed, moved, or renamed.

Determines if the control auto updates files/folders if the user changes the folder structure in the background. If the AutoUpdate property is True, the control receives notification messages each time the folder structure is changed - for example, in another Explorer window that is running. If the user or another program changes the folder structure , the control will update itself accordingly. If the AutoUpdate property is False, the changes in the folder will be updated next time when the control is refreshed. Call the Refresh method to update the control's content.

property CascadeFile.BackColor as Color

Retrieves or sets the control's background.

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

Use the BackColor / BackColorAlternate property to specify the control's background color. The Background property returns or sets a value that indicates the background color for parts in the control. Use the ForeColor / ForeColorAlternate property to change the control's foreground color. Use the SelForeColor and SelBackColor properties to specify the background and foreground colors for selected items. The HotBackColor and HotForeColor properties to specify the background and foreground colors for the item while the cursor hovers it.

property CascadeFile.BackColorAlternate as Color

Specifies the control's alternate background color.

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

Use the BackColor / BackColorAlternate property to specify the control's background color. The Background property returns or sets a value that indicates the background color for parts in the control. Use the ForeColor property to change the control's foreground color. Use the SelForeColor and SelBackColor properties to specify the background and foreground colors for selected items. The HotBackColor and HotForeColor properties to specify the background and foreground colors for the item while the cursor hovers it.

property CascadeFile.BackColorHeader as Color

Specifies the header's background color.

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

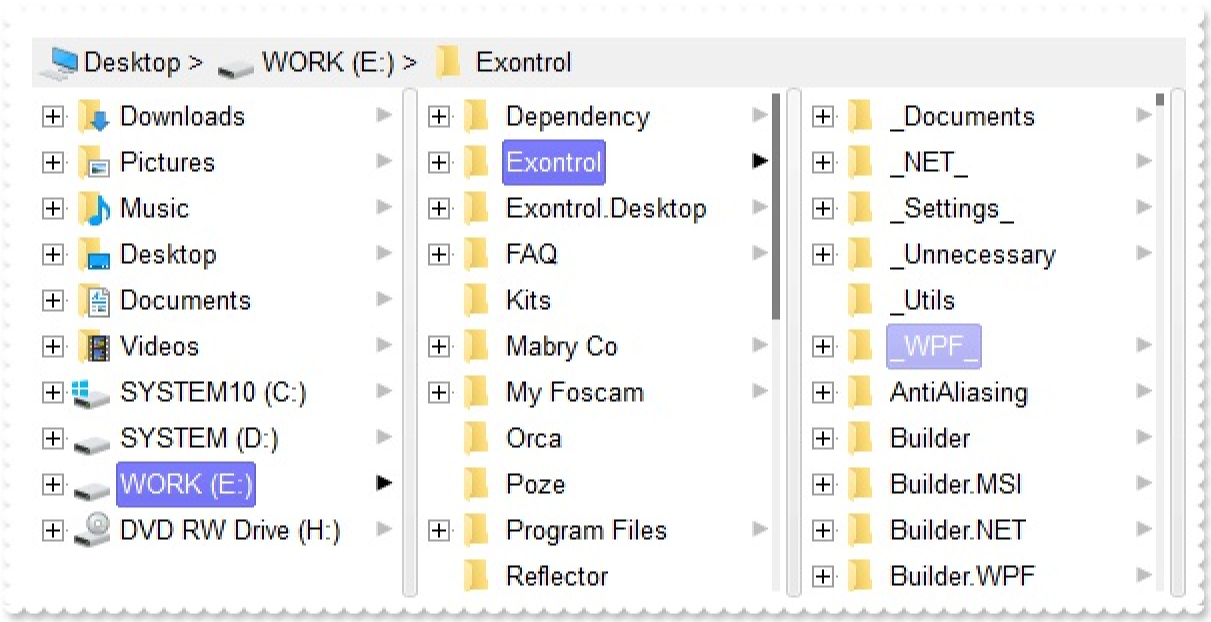
Use the BackColorHeader and ForeColorHeader properties to customize the control's header. Use the HeaderVisible property to hide the control's header. Use the HeaderHeight property to specify the height of the control's header bar. Use the BackColor property to specify the control's background color.

property CascadeFile.Background(Part as BackgroundPartEnum) as Color

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

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

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



method CascadeFile.BeginUpdate ()

Prevents the control from painting until the EndUpdate method is called.

Type	Description

property CascadeFile.BorderHeight as Long

Sets or retrieves a value that indicates the border height of the control.

Type	Description
Long	A long expression that specifies the height of the control's border.

The BorderWidth / BorderHeight property specifies the size of the control's border. The Appearance property retrieves or sets the control's appearance. The Add method to add new skins to the control. Use the BackColor property to specify the control's background color.

property CascadeFile.BorderWidth as Long

Sets or retrieves a value that indicates the border width of the control.

Type	Description
Long	A Long expression that indicates the border width of the control.

By default, the BorderWidth property is 0. The BorderWidth / BorderHeight property specifies the size of the control's border. The Appearance property retrieves or sets the control's appearance. The Add method to add new skins to the control. Use the BackColor property to specify the control's background color.

property CascadeFile.ColumnAutoResize as Boolean

Returns or sets a value indicating whether the control will automatically size its visible columns to fit on the control's client width.

Type	Description
Boolean	A boolean expression indicating whether the control will automatically size its visible columns to fit on the control's client width.

By default, the ColumnAutoResize property is True. If the ColumnAutoResize is True the control has no horizontal scroll bar. If the ColumnAutoResize property is False, the horizontal scroll bar of the control is visible if the sum of visible column's width exceeds the width of the control's client area. Use the ColumnsVisible property to hide a column. Use the HeaderVisible property to show or hide the control's header bar.

property CascadeFile.ColumnsVisible as FileColumnEnum

Indicates the columns being visible.

Type	Description
FileColumnEnum	A FileColumnEnum expression that specifies the columns being visible.

By default, the ColumnsVisible property is exFileColumnName | exFileColumnSize | exFileColumnType | exFileColumnModified (30). The ColumnsVisible property indicates the columns being visible. You can use the ColumnsVisible property to show/hide multiple columns at once. The StatusBarVisible property specifies whether the control's status bar is visible or hidden. The StatusBarLabel property specifies the HTML label the control's status bar is displaying.

property CascadeFile.DefaultItemHeight as Long

Retrieves or sets a value that indicates the default item height.

Type	Description
Long	A long expression indicates the default item height.

By default, the DefaultItemHeight property is 18 pixels. The DefaultItemHeight property specifies the height of the items. The ImageSize property specifies the size of the icons the control displays.

property CascadeFile.DefColumnWidth as Long

Specifies the width to create a new cascade column.

Type	Description
Long	A long expression that specifies the width to create a new cascade column.

By default, the DefColumnWidth property is 256. The DefColumnWidth property specifies the width to create a new cascade column. The Mode property indicates the mode the control displays the cascade columns. The FitCascadeColumns property retrieves or sets a value that indicates the number of cascading columns to fit. The FitToClient method resizes or/and moves the all cascade columns to fit the control's client area.

The following properties can be used to limit / range the width of each cascade columns:

- The MinColumnWidth property specifies the minimum width for any cascade column.
- The MaxColumnWidth property specifies the maximum width for any cascade column.

property CascadeFile.Enabled as Boolean

Enables or disables the control.

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

Use the Enabled property to disable the control. Use the ForeColor property to specify the control's foreground color. Use the BackColor property to specify the control's background color. Use the Font property to specify the control's font.

The following VB sample disables the control:

```
CascadeFile1.Enabled = False
```

The following C++ sample disables the control:

```
m_cascadefile.SetEnabled( FALSE );
```

The following VB.NET sample disables the control:

```
AxCascadeFile1.Enabled = False
```

The following C# sample disables the control:

```
axCascadeFile1.Enabled = false;
```

The following VFP sample disables the control:

```
With thisform.CascadeFile1
   .Object.Enabled = False
EndWith
```

method CascadeFile.EndUpdate ()

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

Type	Description

property CascadeFile.EventParam(Parameter as Long) as Variant

Retrieves or sets a value that indicates the current's event parameter.

Type	Description
Parameter as Long	A long expression that indicates the index of the parameter being requested ie 0 means the first parameter, 1 means the second, and so on. If -1 is used the EventParam property retrieves the number of parameters. Accessing an not-existing parameter produces an OLE error, such as invalid pointer (E_POINTER)
Variant	A VARIANT expression that specifies the parameter's value.

The EventParam method is provided to allow changing the event's parameters passed by reference, even if your environment does not support changing it (uniPaas 1.5 (formerly known as eDeveloper), DBase, and so on). For instance, Unipaas event-handling logic cannot update ActiveX control variables by updating the received arguments. The EventParam(0) retrieves the value of the first parameter of the event, while the EventParam(1) = 0, changes the value of the second parameter to 0 (the operation is successfully, only if the parameter is passed by reference). The EventParam(-1) retrieves the number of the parameters of the current event.

Let's take the event "event KeyDown (**KeyCode** as Integer, ByVal Shift as Integer)", where the KeyCode parameter is passed by reference. For instance, put the KeyCode parameter on 0, and the arrow keys are disabled while the control has the focus.

In most languages you will type something like:

```
Private Sub Control1_KeyDown(KeyCode As Integer, Shift As Integer)
   KeyCode = 0
End Sub
```

In case your environment does not support events with parameters by reference, you can use a code like follows:

```
Private Sub Control1_KeyDown(KeyCode As Integer, Shift As Integer)
   Control1.EventParam(0) = 0
End Sub
```

In other words, the EventParam property provides the parameters of the current event for reading or writing access, even if your environment does not allow changing parameters by reference.

Calling the EventParam property outside of an event produces an OLE error, such as pointer invalid, as its scope was designed to be used only during events.

method CascadeFile.ExecuteContextCommand (FileName as String, Folder as Boolean, Command as String)

Executes a context menu command.

Type	Description
FileName as String	A string expression that indicates the file whose context menu is invoked.
Folder as Boolean	A boolean expression that indicates whether the FileName points to a folder or to a file.
Command as String	A string expression that indicates the name of the command being invoked or a string expression that indicates the identifier of command being invoked.

The ExecuteContextCommand method executes the object's context menu command. The control returns no error, if the command is not found. Use the AllowContextMenu property to disable the control's context menu when user right clicks a file in the browser.

Here's the list of the identifiers for some known items in the object's context menu :

- Create Shortcut **(17)**
- Delete **(18)**
- Properties **(20)**
- Cut **(25)**
- Copy **(26)**

property CascadeFile.ExecuteContextMenu as Long

Executes a command from the object's context menu.

Type	Description
Long	A Long expression that determines the identifier of the command to be executed.

By default, the ExecuteContextMenu property is 0. The ExecuteContextMenu property specifies the identifier of the command to be executed (id option in the ShowContextMenu property). The ExecuteContextMenu property has effect only during the ViewEndChanging event, when the Operation parameter is exExecuteContextMenu(21). The AllowContextMenu property specifies whether the control shows the object's context menu when the user presses the right click over a file or folder.

The following sample shows how you can append new items to the object's context menu and displays a message when a command is selected from the context menu:

```
Private Sub CascadeFile1_StateChange(ByVal State As EXMILLERLibCtl.StateChangeEnum)
    With CascadeFile1
        If (State = ShowContextMenu) Then
            .ShowContextMenu = .ShowContextMenu + ",Item 1[id=1][def],Popup[id=2](Sub-
Item 2[id=2],[sep],Sub-Item 3[id=3])"
        Else
            If (State = ExecuteContextMenu) Then
                Debug.Print "You selected the command: " & .ExecuteContextMenu
            End If
        End If
    End With
End Sub
```

The following sample shows how you can prevent executing a specific command:

```
Private Sub CascadeFile1_StateChange(ByVal State As EXMILLERLibCtl.StateChangeEnum)
    With CascadeFile1
        If (State = ExecuteContextMenu) Then
            If Not (.ExecuteContextMenu = 17) Then   ' Delete
                Debug.Print "You selected the command: " & .ExecuteContextMenu
            Else
                .ExecuteContextMenu = 0
                MsgBox "Delete is disabled."
            End If
        End If
    End With
End Sub
```

method CascadeFile.ExecuteTemplate (Template as String)

Executes a template and returns the result.

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

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

For instance, the following sample retrieves the beginning date (as string) for the default bar in the first visible item:

```
Debug.Print CascadeFile1.ExecuteTemplate("Items.ItemBar(FirstVisibleItem(),``,1)")
```

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

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

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

The Template script is composed by lines of instructions. Instructions are separated by

"\n\r" (newline) characters.

An instruction can be one of the following:

- **Dim** list of variables *Declares the variables. Multiple variables are separated by commas. (Sample: Dim h, h1, h2)*
- variable **=** property(list of arguments) *Assigns the result of the property to a variable. The "variable" is the name of a declared variable. The "property" is the property name of the object in the context. The "list or arguments" may include variables or values separated by commas. (Sample: h = InsertItem(0,"New Child"))*
- property**(** list of arguments **)** = value *Changes the property. The value can be a variable, a string, a number, a boolean value or a RGB value.*
- method**(** list of arguments **)** *Invokes the method. The "list or arguments" may include variables or values separated by commas.*
- **{** *Beginning the object's context. The properties or methods called between { and } are related to the last object returned by the property prior to { declaration.*
- **}** *Ending the object's context*
- object**.** property(list of arguments)**.**property(list of arguments).... *The .(dot) character splits the object from its property. For instance, the Columns.Add("Column1").HeaderBackColor = RGB(255,0,0), adds a new column and changes the column's header back color.*

The Template supports the following general functions:

- **RGB(**R,G,B**)** *property retrieves an RGB value, where the R, G, B are byte values that indicates the R G B values for the color being specified. For instance, the following code changes the control's background color to red: BackColor = RGB(255,0,0)*
- **CreateObject(**progID**)** *property creates and retrieves a single uninitialized object of the class associated with a specified program identifier.*

property CascadeFile.ExpandFolders as Boolean

Retrieves or sets a value that indicates whether the control expands the folder objects.

Type	Description
Boolean	A boolean expression that indicates whether the control expands the folder objects.

By default, the ExpandFolders property is True. If the ExpandFolders property is True, each folder that contains a subfolder, displays +/- button, that allows to expand or collapse the folder. You can use the ExpandFolder property to let CascadeFile control simulates a folderview control. Use the ExploreFromHere property specifies the root folder for the control.

property CascadeFile.ExploreFromHere as String

Specifies the root folder for the control.

Type	Description
String	A string expression that indicates the folder's path that's the root of the control. The control can display multiple root items, separated by | character. For instance, "C:\\|D:\" specifies that the control should display the C drive and D as well.

By default, the ExploreFromHere property is "". The ExplorerFromHere property specifies the root folder for the control. The ExpandFolders property retrieves or sets a value that indicates whether the control expands the folder objects.

property CascadeFile.FitCascadeColumns as Long

Retrieves or sets a value that indicates the number of cascading columns to fit.

Type	Description
Long	A long expression that indicates the number of cascading columns to fit.

By default, the FitCascadeColumns property is 4. The FitCascadeColumns property retrieves or sets a value that indicates the number of cascading columns to fit. The FitToClient method resizes or/and moves the all cascade columns to fit the control's client area. The DefColumnWidth property specifies the width to create a new cascade column. The Mode property indicates the mode the control displays the cascade columns.

The following properties can be used to limit / range the width of each cascade columns:

- The MinColumnWidth property specifies the minimum width for any cascade column.
- The MaxColumnWidth property specifies the maximum width for any cascade column.

method CascadeFile.FitToClient ([FitColumnsCount as Variant])

Resizes or/and moves the all cascade columns to fit the control's client area.

Type	Description
FitColumnsCount as Variant	A long expression that specifies the number of columns to fit. If missing, it indicates 4.

The FitToClient method resizes or/and moves the all cascade columns to fit the control's client area. The DefColumnWidth property specifies the width to create a new cascade column. The Mode property indicates the mode the control displays the cascade columns. The FitCascadeColumns property retrieves or sets a value that indicates the number of cascading columns to fit.

The following properties can be used to limit / range the width of each cascade columns:

- The MinColumnWidth property specifies the minimum width for any cascade column.
- The MaxColumnWidth property specifies the maximum width for any cascade column.

property CascadeFile.Font as IFontDisp

Retrieves or sets the Font object used to paint control.

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

Use the Font property to change the control's font. Use the Refresh method to refresh the control.

The following VB sample assigns by code a new font to the control:

```
With CascadeFile1
    With .Font
        .Name = "Tahoma"
    End With
    .Refresh
End With
```

The following C++ sample assigns by code a new font to the control:

```
COleFont font = m_cascadefile.GetFont();
font.SetName( "Tahoma" );
m_cascadefile.Refresh();
```

the C++ sample requires definition of COleFont class (#include "Font.h")

The following VB.NET sample assigns by code a new font to the control:

```
With AxCascadeFile1
    Dim font As System.Drawing.Font = New System.Drawing.Font("Tahoma", 10,
FontStyle.Regular, GraphicsUnit.Point)
    .Font = font
    .CtlRefresh()
End With
```

The following C# sample assigns by code a new font to the control:

```
System.Drawing.Font font = new System.Drawing.Font("Tahoma", 10, FontStyle.Regular);
axCascadeFile1.Font = font;
```

```
axCascadeFile1.CtlRefresh();
```

The following VFP sample assigns by code a new font to the control:

```
with thisform.CascadeFile1.Object
    .Font.Name = "Tahoma"
    .Refresh()
endwith
```

property CascadeFile.ForeColor as Color

Retrieves or sets the control's foreground color.

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

Use the ForeColor / ForeColorAlternate property to change the control's foreground color. Use the BackColor / BackColorAlternate property to specify the control's background color. The Background property returns or sets a value that indicates the background color for parts in the control. Use the SelForeColor and SelBackColor properties to specify the background and foreground colors for selected items. The HotBackColor and HotForeColor properties to specify the background and foreground colors for the item while the cursor hovers it.

property CascadeFile.ForeColorAlternate as Color

Specifies the control's alternate foreground color.

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

Use the ForeColor / ForeColorAlternate property to change the control's foreground color. Use the BackColor / BackColorAlternate property to specify the control's background color. The Background property returns or sets a value that indicates the background color for parts in the control. Use the SelForeColor and SelBackColor properties to specify the background and foreground colors for selected items. The HotBackColor and HotForeColor properties to specify the background and foreground colors for the item while the cursor hovers it.

property CascadeFile.ForeColorHeader as Color

Specifies the header's foreground color.

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

Use the BackColorHeader and ForeColorHeader properties to customize the control's header. Use the HeaderVisible property to hide the control's header. Use the HeaderHeight property to specify the height of the control's header bar. Use the ForeColor property to specify the control's foreground color.

method CascadeFile.FormatABC (Expression as String, [A as Variant], [B as Variant], [C as Variant])

Formats the A,B,C values based on the giving expression and returns the result.

Type	Description
Expression as String	A String that defines the expression to be evaluated.
A as Variant	A VARIANT expression that indicates the value of the A keyword.
B as Variant	A VARIANT expression that indicates the value of the B keyword.
C as Variant	A VARIANT expression that indicates the value of the C keyword.

Return	Description
Variant	A VARIANT expression that indicates the result of the evaluation the CascadeFile.

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

For instance:

- "A + B + C", adds / concatenates the values of the A, B and C
- "value MIN 0 MAX 99", limits the value between 0 and 99
- "value format ``", formats the value with two decimals, according to the control's panel setting
- "date(`now`)" returns the current time as double

The FormatABC method supports the following keywords, constants, operators and functions:

- **A** or **value** keyword, indicates a variable A whose value is giving by the A parameter
- **B** keyword, indicates a variable B whose value is giving by the B parameter
- **C** keyword, indicates a variable C whose value is giving by the C parameter

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

property CascadeFile.FormatAnchor(New as Boolean) as String

Specifies the visual effect for anchor elements in HTML captions.

Type	Description
New as Boolean	Boolean expression that indicates whether to specify the anchors never clicked or anchors being clicked.
String	A String expression that indicates the HTMLformat to apply to anchor elements.

By default, the FormatAnchor(**True**) property is "<u><fgcolor=0000FF>#" that indicates that the anchor elements (that were never clicked) are underlined and shown in light blue. Also, the FormatAnchor(**False**) property is "<u><fgcolor=000080>#" that indicates that the anchor elements are underlined and shown in dark blue. The visual effect is applied to the anchor elements, if the FormatAnchor property is not empty. For instance, if you want to do not show with a new effect the clicked anchor elements, you can use the FormatAnchor(**False**) = "", that means that the clicked or not-clicked anchors are shown with the same effect that's specified by FormatAnchor(**True**). An anchor is a piece of text or some other object (for example an image) which marks the beginning and/or the end of a hypertext link. The **<a>** element is used to mark that piece of text (or inline image), and to give its hypertextual relationship to other documents. The control fires the AnchorClick event to notify that the user clicks an anchor element. This event is fired only if prior clicking the control it shows the hand cursor. The AnchorClick event carries the identifier of the anchor, as well as application options that you can specify in the anchor element. The hand cursor is shown when the user hovers the mouse on the anchor elements.

method CascadeFile.FreezeEvents (Freeze as Boolean)

Prevents the control to fire any event.

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

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

property CascadeFile.FullRowSelect as Boolean

Enables full-row selection in the control.

Type	Description
Boolean	A Boolean expression that specifies how the selection is shown in the control.

By default, the FullRowSelect property is False, which indicates that the Name column shows the selected filed. If the FullRowSelect property is True, the entire item (Name, Size, Type and Modified columns) is shown as selected. The SingleSel property indicates whether the control supports single or multiple selection. Use the SelForeColor property to specify the foreground color for selected files or folders. Use the SelBackColor property to specify the background color for selected files or folders.

property CascadeFile.Get (Type as TypeEnum) as Variant

Gets the list of files (selected, checked, visible, ...) as a safe array of VARIANT.

Type	Description
Type as TypeEnum	A TypeEnum expression that indicates the type of files to retrieve
Variant	A safe array of VARIANT that includes the files

The Get method returns all, visible, selected or checked files. Use the HasCheckBox property to assign a check box for each item in your control. Use the Get method to retrieve :

- selected files and folders
- all items in the control
- checked files, folders
- the list of visible items, as they are displayed

The following VB sample enumerates the files being checked:

```
Dim f
For Each f In CascadeFile1.Get(CheckItems)
    Debug.Print f
Next
```

The following VB/NET sample enumerates the files being checked:

```
For Each f In Excascadefile1.get_Get(exontrol.EXMILLERLib.TypeEnum.CheckItems)
    Debug.Print(f)
Next
```

The following C# sample enumerates the files being checked:

```
foreach (var file in excascadefile1.get_Get(exontrol.EXMILLERLib.TypeEnum.CheckItems))
    System.Diagnostics.Debug.Print(file.ToString());
```

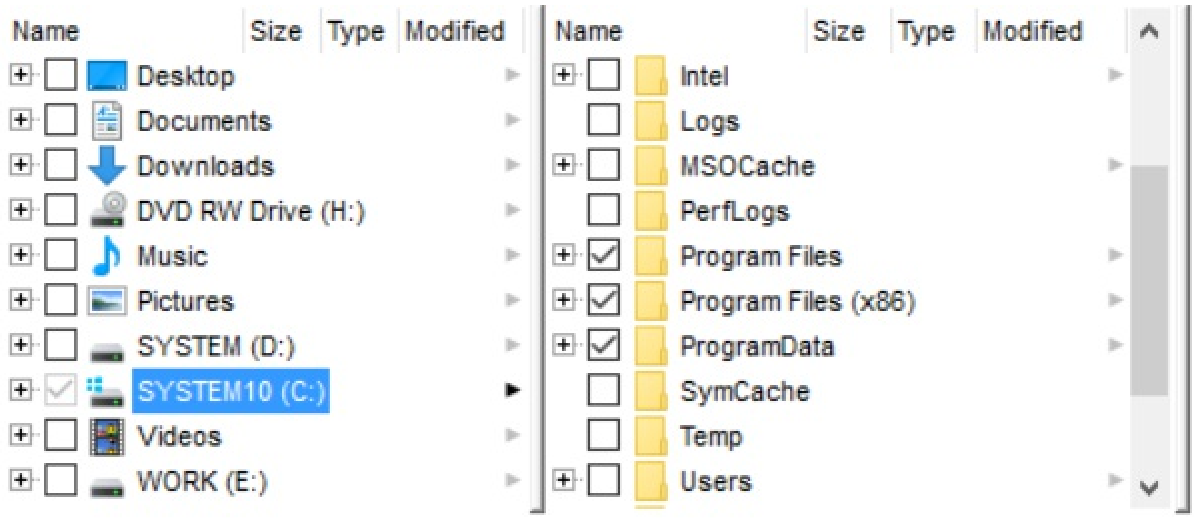
property CascadeFile.HasCheckBox as CheckBoxEnum

Specifies whether the control displays a check box for each item.

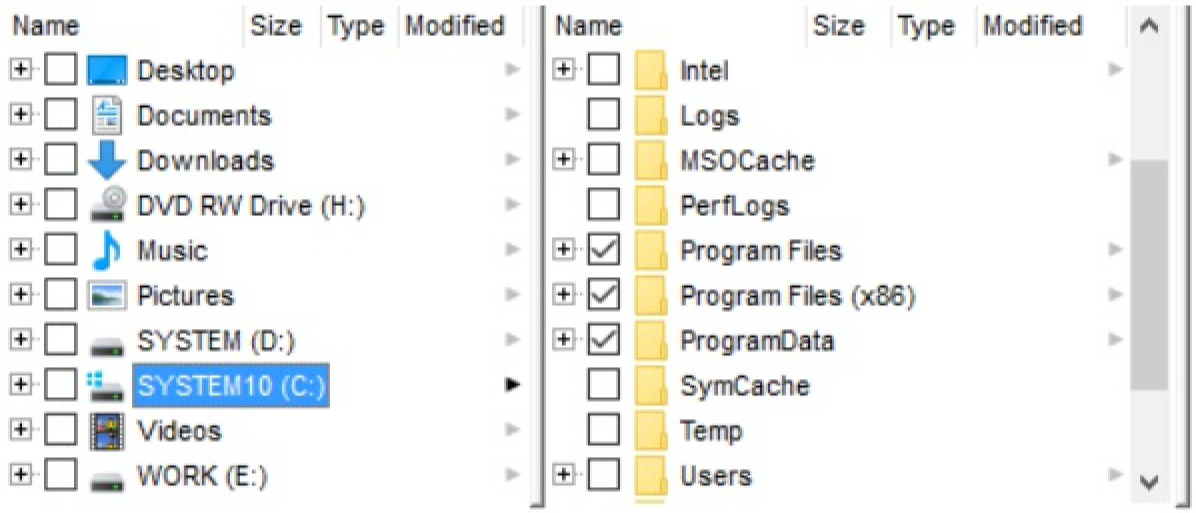
Type	Description
CheckBoxEnum	A CheckBoxEnum expression that specifies whether the control displays check boxes

By default, the HasCheckBox property is NoCheckBox, which indicates that the control display no check boxes. Use the HasCheckBox property to add check or partial-check support to the control. The ViewStartChanging(exCheckStateChange) and ViewEndChanging(exCheckStateChange) events notify once the user changes the file's check box. The Get(CheckItems) property returns the files being checked.

The following screen shot shows the control with partial-check support:



The following screen shot shows the control with check support:



property CascadeFile.HeaderAppearance as AppearanceEnum

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

Type	Description
AppearanceEnum	An AppearanceEnum expression that indicates the header's appearance.

Use the HeaderAppearance property to change the appearance of the control's header bar. Use the Appearance property to specify the control's appearance. Use the HeaderVisible property to hide the control's header bar. The HeaderHeight property retrieves or sets a value indicating the control's header height. Use the ColumnsVisible property to hide or show a column. Use the BackColorHeader and ForeColorHeader properties to customize the control's header. Use the BackColor property to specify the control's background color. The StatusBarVisible property specifies whether the control's status bar is visible or hidden. The StatusBarLabel property specifies the HTML label the control's status bar is displaying.

property CascadeFile.HeaderHeight as Long

Retrieves or sets a value indicating the control's header height.

Type	Description
Long	A long expression that specifies the control's header height

By default, the HeaderHeight property is 18 pixels. The HeaderHeight property retrieves or sets a value indicating the control's header height. Use the HeaderVisible property to hide the control's header bar. Use the ColumnsVisible property to hide or show a column. Use the BackColorHeader and ForeColorHeader properties to customize the control's header. Use the BackColor property to specify the control's background color. The StatusBarVisible property specifies whether the control's status bar is visible or hidden. The StatusBarLabel property specifies the HTML label the control's status bar is displaying.

property CascadeFile.HeaderVisible as Boolean

Retrieves or sets a value that indicates whether the control's header bar is visible or hidden.

Type	Description
Boolean	A boolean expression that indicates whether the control's header bar is visible or hidden.

Use the HeaderVisible property to hide the control's header bar. The HeaderHeight property retrieves or sets a value indicating the control's header height. Use the ColumnsVisible property to hide or show a column. Use the BackColorHeader and ForeColorHeader properties to customize the control's header. Use the BackColor property to specify the control's background color. The StatusBarVisible property specifies whether the control's status bar is visible or hidden. The StatusBarLabel property specifies the HTML label the control's status bar is displaying.

property CascadeFile.HideSelection as Boolean

Returns a value that determines whether selected item appears highlighted when a control loses the focus.

Type	Description
Boolean	A boolean expression that indicates whether the selected item appears highlighted when a control loses the focus.

By default, the HideSelection property is False. You can use this property to indicate which item is highlighted while another form or a dialog box has the focus. Use the SelForeColor and SelBackColor property to customize the colors for the selected items in the control.

property CascadeFile.HotBackColor as Color

Retrieves or sets a value that indicates the hot-tracking background color.

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

Use the HotBackColor property on a non-zero value to highlight the item from the cursor. The HotBackColor property has effect while it is not -1. For instance, you can set the HotBackColor property on -1, and so no item will be highlighted while the cursor hovers it. The HotForeColor property specifies the foreground color to highlight the item from the cursor. The SelBackColor property specifies the selection background color.

property CascadeFile.HotForeColor as Color

Retrieves or sets a value that indicates the hot-tracking foreground color.

Type	Description
Color	A color expression that indicates the foreground color for item from the cursor (hovering the item).

Use the HotForeColor property on a non-zero value to highlight the item from the cursor. The HotForeColor property has effect while it is not -1. For instance, you can set the HotForeColor property on -1, and so no item will be highlighted while the cursor hovers it. The HotBackColor property specifies the background color to highlight the item from the cursor. The SelForeColor property specifies the selection foreground color.

property CascadeFile.HTMLPicture(Key as String) as Variant

Adds or replaces a picture in HTML captions.

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

The HTMLPicture property handles a collection of custom size picture being displayed in the HTML captions, using the <img> tags. By default, the HTMLPicture collection is empty. Use the HTMLPicture property to add new pictures to be used in HTML captions. For instance, the HTMLPicture("pic1") = "c:\winnt\zapotec.bmp", loads the zapotec picture and associates the pic1 key to it. Any "<img>pic1</img>" sequence in HTML captions, displays the pic1 picture. On return, the HTMLPicture property retrieves a Picture object (this implements the IPictureDisp interface).

The following sample shows how to put a custom size picture in the column's header:

```
<CONTROL>.HTMLPicture("pic1") = "c:/temp/editors.gif"
<CONTROL>.HTMLPicture("pic2") = "c:/temp/editpaste.gif"

<COLUMN1>.HTMLCaption = "A <img>pic1</img>"
<COLUMN2>.HTMLCaption = "B <img>pic2</img>"
<COLUMN3>.HTMLCaption = "A <img>pic1</img> + B <img>pic2</img>"
```

property CascadeFile.hWnd as Long

Retrieves the control's window handle.

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

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

method CascadeFile.Images (Handle as Variant)

Sets the control's image list at runtime.

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

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

The user can add images at design time, by drag and drop files to combo's image holder. The ImageSize property specifies the size of the icons the control displays. Use the ReplaceIcon method to add, remove or clear icons in the control's images collection.

property CascadeFile.ImageSize as Long

Retrieves or sets the size of icons the control displays.

Type	Description
Long	A long expression that defines the size of icons the control displays.

By default, the ImageSize property is 16 (pixels). The ImageSize property specifies the size of the icons the control displays. Use the Images method to attach a image list to the control. The DefaultItemHeight property specifies the height of the items.

The ImageSize property defines the size to display the following UI elements:

- any icon that a cell or column displays
- check-box or radio-buttons
- expand/collapse glyphs
- header's sorting or drop down-filter glyphs

property CascadeFile.Indent as Long

Retrieves or sets the amount, in pixels, that child items are indented relative to their parent items.

Type	Description
Long	A long expression that indicates the amount, in pixels, that child items are indented relative to their parent items.

By default, the Indent property is 12 pixels. The Indent property specifies the amount, in pixels, that child items are indented relative to their parent items. If the Indent property is 0, the child items are not indented relative to their parent item.

property CascadeFile.Layout as String

Saves or loads the control's layout, such current selection for each panel, the widths of the cascade columns, and so on.

Type	Description
String	A String expression that defines the current's layout.

The Layout property saves or loads the control's layout, such current selection for each panel, the widths of the cascade columns, and so on. For instance, you can use the Layout property to save and restore later the current views, which includes the selection in each panel, and so on.

property CascadeFile.MaxColumnWidth as Long

Specifies the maximum width for any cascade column.

Type	Description
Long	A Long expression that specifies the maximum width for any cascade column.

By default, the MaxColumnWidth property is -1. While negative, there is no upper limit, so the MaxColumnWidth property has no effect. The Mode property indicates the mode the control displays the cascade columns. The DefColumnWidth property specifies the width to create a new cascade column. The FitCascadeColumns property retrieves or sets a value that indicates the number of cascading columns to fit. The FitToClient method resizes or/and moves the all cascade columns to fit the control's client area.

The following properties can be used to limit / range the width of each cascade columns:

- The MinColumnWidth property specifies the minimum width for any cascade column.
- The MaxColumnWidth property specifies the maximum width for any cascade column.

property CascadeFile.MinColumnWidth as Long

Specifies the minimum width for any cascade column.

Type	Description
Long	A Long expression that specifies the minimum width for any cascade column.

By default, the MinColumnWidth property is 32. The Mode property indicates the mode the control displays the cascade columns. The DefColumnWidth property specifies the width to create a new cascade column. The FitCascadeColumns property retrieves or sets a value that indicates the number of cascading columns to fit. The FitToClient method resizes or/and moves the all cascade columns to fit the control's client area.

The following properties can be used to limit / range the width of each cascade columns:

- The MinColumnWidth property specifies the minimum width for any cascade column.
- The MaxColumnWidth property specifies the maximum width for any cascade column.

property CascadeFile.Mode as CascadeModeEnum

Indicates the mode the control displays the cascade columns.

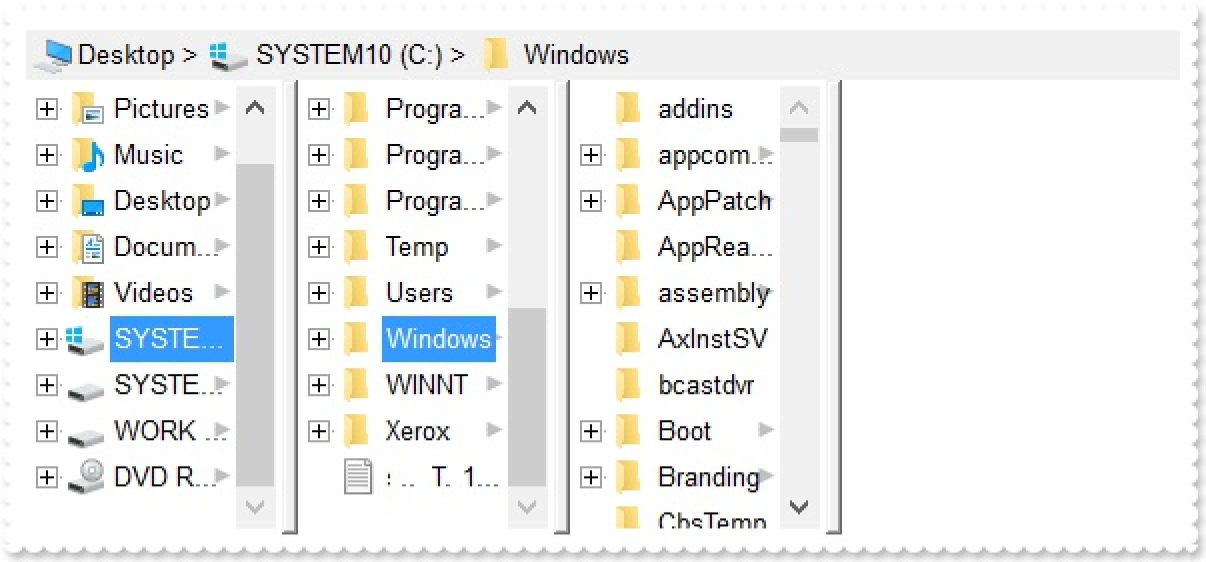
Type	Description
CascadeModeEnum	A CascadeModeEnum expression that indicates how the control displays the cascade columns once a file or more are selected.

By default, the Mode property is exSplitFixCascadeMode | exAutoFitOnResizeClient. The Mode property indicates the mode the control displays the cascade columns. The DefColumnWidth property specifies the width to create a new cascade column. The FitCascadeColumns property retrieves or sets a value that indicates the number of cascading columns to fit. The FitToClient method resizes or/and moves the all cascade columns to fit the control's client area.

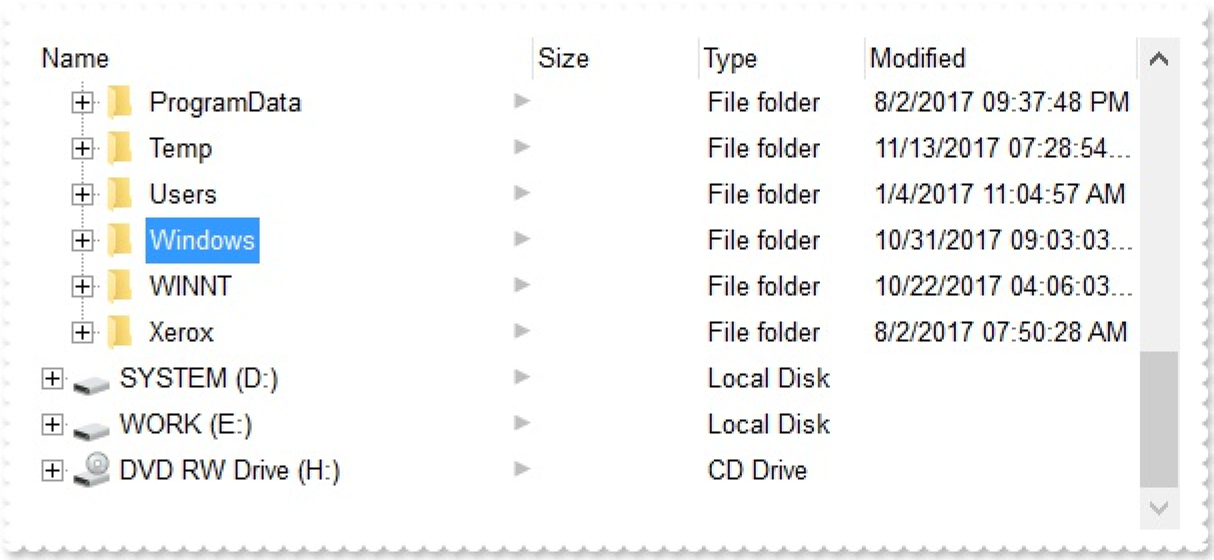
The following properties can be used to limit / range the width of each cascade columns:

- The MinColumnWidth property specifies the minimum width for any cascade column.
- The MaxColumnWidth property specifies the maximum width for any cascade column.

The following screen shot shows the control while the Mode property includes exFixCascadeMode:

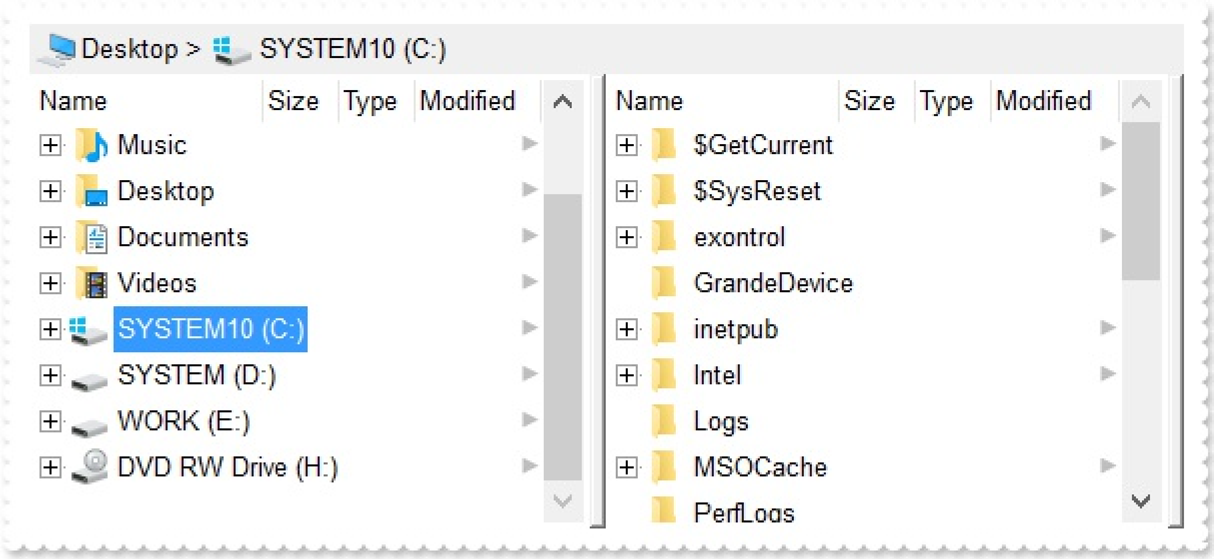


The following screen shot shows the control while the Mode property includes exSingleCascadeMode:



Name	Size	Type	Modified
ProgramData		File folder	8/2/2017 09:37:48 PM
Temp		File folder	11/13/2017 07:28:54...
Users		File folder	1/4/2017 11:04:57 AM
Windows		File folder	10/31/2017 09:03:03...
WINNT		File folder	10/22/2017 04:06:03...
Xerox		File folder	8/2/2017 07:50:28 AM
SYSTEM (D:)		Local Disk	
WORK (E:)		Local Disk	
DVD RW Drive (H:)		CD Drive	

The following screen shot shows the control while the Mode property includes exSplitEqualCascadeMode:

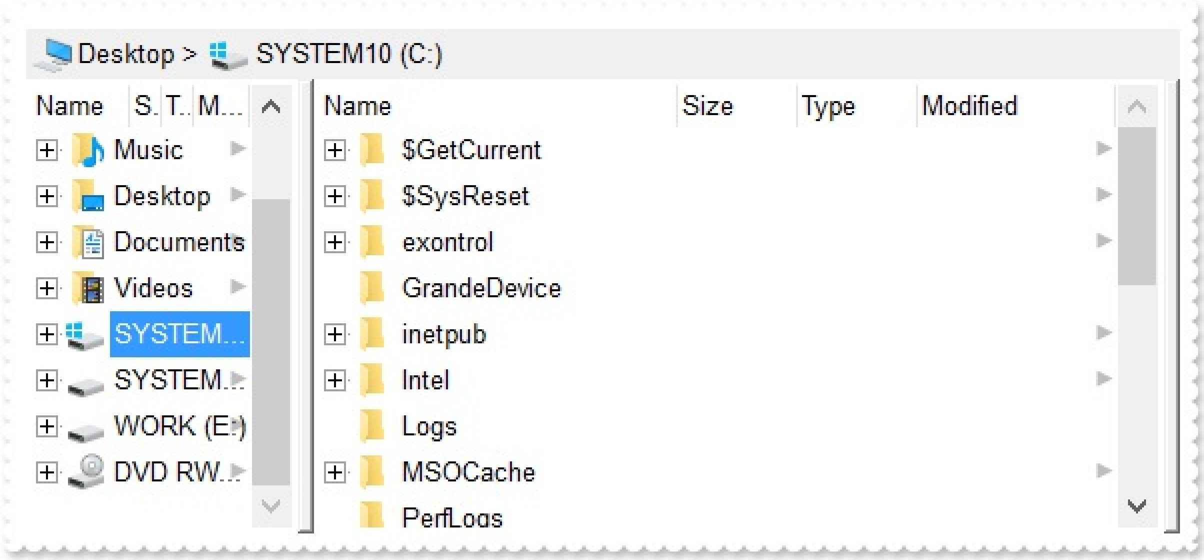


Desktop > SYSTEM10 (C:)

Name	Size	Type	Modified
Music			
Desktop			
Documents			
Videos			
SYSTEM10 (C:)			
SYSTEM (D:)			
WORK (E:)			
DVD RW Drive (H:)			

Name	Size	Type	Modified
$GetCurrent			
$SysReset			
exontrol			
GrandeDevice			
inetpub			
Intel			
Logs			
MSOCache			
PerfLogs			

The following screen shot shows the control while the Mode property includes exSplitFixCascadeMode:



Desktop > SYSTEM10 (C:)

Name	S.	T..	M...
Music			
Desktop			
Documents			
Videos			
SYSTEM...			
SYSTEM...			
WORK (E:)			
DVD RW...			

Name	Size	Type	Modified
$GetCurrent			
$SysReset			
exontrol			
GrandeDevice			
inetpub			
Intel			
Logs			
MSOCache			
PerfLogs			

property CascadeFile.Picture as IPictureDisp

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

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

Use the Picture property to load a picture on the control's background. By default, the control has no picture associated. Use the PictureDisplay property to layout the control's picture on the control's background. Use the BackColor property to specify the control's background color. Use the ForeColor property to change the control's foreground color. Use the SelForeColor and SelBackColor properties to specify the background and foreground colors for selected items.

property CascadeFile.PictureDisplay as PictureDisplayEnum

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

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

By default, the PictureDisplay property is exTile. Use the PictureDisplay property specifies how the Picture is displayed on the control's background. If the control has no picture associated the PictureDisplay property has no effect. Use the BackColor property to specify the control's background color. Use the ForeColor property to change the control's foreground color. Use the SelForeColor and SelBackColor properties to specify the background and foreground colors for selected items

method CascadeFile.Refresh ()

Refreshes the control.

Type	Description

The Refresh method refreshes the control's content.

The following VB sample calls the Refresh method:

```
CascadeFile1.Refresh
```

The following C++ sample calls the Refresh method:

```
m_cascadefile.Refresh();
```

The following VB.NET sample calls the Refresh method:

```
AxCascadeFile1.CtlRefresh()
```

In VB.NET the System.Windows.Forms.Control class has already a Refresh method, so the CtlRefresh method should be called.

The following C# sample calls the Refresh method:

```
axCascadeFile1.CtlRefresh();
```

In C# the System.Windows.Forms.Control class has already a Refresh method, so the CtlRefresh method should be called.

The following VFP sample calls the Refresh method:

```
thisform.CascadeFile1.Object.Refresh()
```

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

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

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

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

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

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

i = CascadeFile1.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 = CascadeFile1.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:

CascadeFile1.ReplaceIcon 0, i, where i specifies the index of icon removed.

The following VB clears the control's icons collection:

CascadeFile1.ReplaceIcon 0, -1

property CascadeFile.ScrollButtonHeight as Long

Specifies the height of the button in the vertical scrollbar.

Type	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 ScrollButtonWidth property to specify the width of the buttons in the horizontal scroll bar. Use the ScrollWidth property to specify the width of the vertical 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 control's scroll bar. Use the ScrollThumbSize property to define a fixed size for the scrollbar's thumb.

property CascadeFile.ScrollButtonWidth as Long

Specifies the width of the button in the horizontal scrollbar.

Type	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 ScrollButtonHeight property to specify the height of the buttons in the vertical scroll bar. Use the ScrollWidth property to specify the width of the vertical 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 control's scroll bar. Use the ScrollThumbSize property to define a fixed size for the scrollbar's thumb.

property CascadeFile.ScrollFont (ScrollBar as ScrollBarEnum) as IFontDisp

Retrieves or sets the scrollbar's font.

Type	Description
ScrollBar as ScrollBarEnum	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 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.

property CascadeFile.ScrollHeight as Long

Specifies the height of the horizontal scrollbar.

Type	Description
Long	A long expression that defines the height of the horizontal scroll bar.

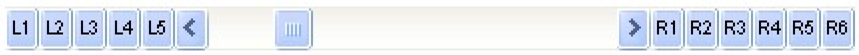
By default, the ScrollHeight property is -1. If the ScrollHeight property is -1, the control uses the default height of the horizontal scroll bar from the system. Use the ScrollHeight property to specify the height of the horizontal scroll bar. Use the ScrollButtonWidth property to specify the width of the buttons in the horizontal scroll bar. Use the ScrollWidth property to specify the width of the vertical scroll bar. Use the ScrollButtonHeight property to specify the height of the buttons in the vertical scroll bar. Use the ScrollPartVisible property to specify the visible parts in the control's scroll bar. Use the ScrollThumbSize property to define a fixed size for the scrollbar's thumb.

property CascadeFile.ScrollOrderParts(ScrollBar as ScrollBarEnum) as String

Specifies the order of the buttons in the scroll bar.

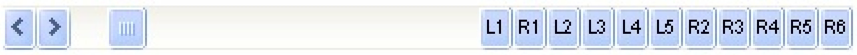
Type	Description
ScrollBar as ScrollBarEnum	A ScrollBar expression that indicates the scrollbar where the order of buttons is displayed.
String	A String expression that indicates the order of the buttons in the scroll bar. The list includes expressions like l, l1, ..., l5, t, r, r1, ..., r6 separated by comma, each expression indicating a part 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:



so, the order of the parts is: l1, l2, l3, l4, l5, l, t, r, r1, r2, r3, r4, r5 and r6. Use the ScrollPartVisible to specify whether a button in the scrollbar is visible or hidden. Use the ScrollPartEnable property to enable or disable a button in the scroll bar. Use the ScrollPartCaption 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, "l,r,t,l1,r1" puts the left and right buttons to the left of the thumb area, and the l1 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.



The list of supported literals in the ScrollOrderParts property is:

- **l** for exLeftBPart, (<) The left or top button.
- **l1** for exLeftB1Part, (L1) The first additional button, in the left or top area.
- **l2** for exLeftB2Part, (L2) The second additional button, in the left or top area.
- **l3** for exLeftB3Part, (L3) The third additional button, in the left or top area.
- **l4** for exLeftB4Part, (L4) The forth additional button, in the left or top area.
- **l5** 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.

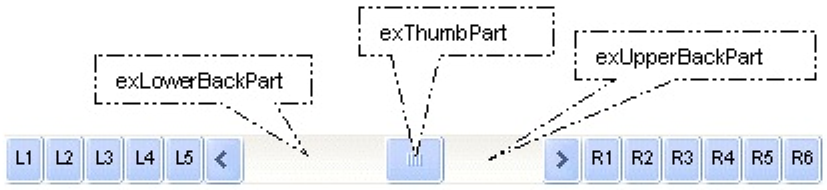
Any other literal between commas is ignored. If duplicate literals are found, the second is ignored, and so on. For instance, "t,l,r" indicates that the left/top and right/bottom buttons are displayed right/bottom after the thumb area.

property CascadeFile.ScrollPartCaption(ScrollBar as ScrollBarEnum, Part as ScrollPartEnum) as String

Specifies the caption being displayed on the specified scroll part.

Type	Description
ScrollBar as ScrollBarEnum	A ScrollBar expression that indicates the scrollbar where the caption is displayed.
Part as ScrollPartEnum	A ScrollPartEnum expression that specifies the parts of the scroll where the text is displated
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 ScrollFont property to specify the font in the control's scroll bar. Use the ScrollOrderParts 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 CascadeFile1
    .BeginUpdate
        .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 AxCascadeFile1
    .BeginUpdate()
    .set_ScrollPartVisible(EXEXMILLERLib.ScrollBarEnum.exVScroll,
EXEXMILLERLib.ScrollPartEnum.exLeftB1Part Or
EXEXMILLERLib.ScrollPartEnum.exRightB1Part, True)
    .set_ScrollPartCaption(EXEXMILLERLib.ScrollBarEnum.exVScroll,
EXEXMILLERLib.ScrollPartEnum.exLeftB1Part, "<img></img>1")
    .set_ScrollPartCaption(EXEXMILLERLib.ScrollBarEnum.exVScroll,
EXEXMILLERLib.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 :

```
axCascadeFile1.BeginUpdate();
axCascadeFile1.set_ScrollPartVisible(EXEXMILLERLib.ScrollBarEnum.exVScroll,
EXEXMILLERLib.ScrollPartEnum.exLeftB1Part |
EXEXMILLERLib.ScrollPartEnum.exRightB1Part, true);
axCascadeFile1.set_ScrollPartCaption(EXEXMILLERLib.ScrollBarEnum.exVScroll,
EXEXMILLERLib.ScrollPartEnum.exLeftB1Part , "<img></img>1");
axCascadeFile1.set_ScrollPartCaption(EXEXMILLERLib.ScrollBarEnum.exVScroll,
EXEXMILLERLib.ScrollPartEnum.exRightB1Part, "<img></img>2");
axCascadeFile1.EndUpdate();
```

The following C++ sample adds up and down additional buttons to the control's vertical scroll bar :

```
m_cascadefile.BeginUpdate();
m_cascadefile.SetScrollPartVisible( 0 /*exVScroll*/, 32768  /*exLeftB1Part*/ | 32
/*exRightB1Part*/, TRUE );
```

```
m_cascadefile.SetScrollPartCaption( 0 /*exVScroll*/, 32768  /*exLeftB1Part*/ , _T("<img>
</img>1") );
m_cascadefile.SetScrollPartCaption( 0 /*exVScroll*/, 32 /*exRightB1Part*/ , _T("<img>
</img>2") );
m_cascadefile.EndUpdate();
```

The following VFP sample adds up and down additional buttons to the control's vertical scroll bar :

```
With thisform.CascadeFile1
    .BeginUpdate
        .ScrollPartVisible(0, bitor(32768,32)) = .t.
        .ScrollPartCaption(0,32768) = "<img></img>1"
        .ScrollPartCaption(0, 32) = "<img></img>2"
    .EndUpdate
EndWith
```

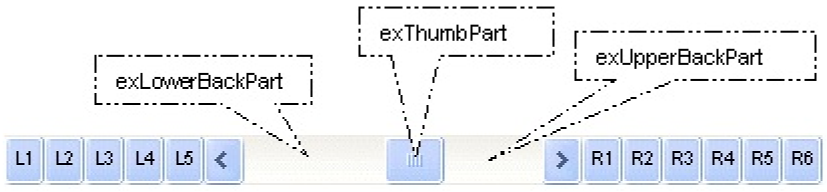
```
*** ActiveX Control Event ***
LPARAMETERS scrollpart

wait window nowait ltrim(str(scrollpart))
```

property CascadeFile.ScrollPartEnable(ScrollBar as ScrollBarEnum, Part as ScrollPartEnum) as Boolean

Indicates whether the specified scroll part is enabled or disabled.

Type	Description
ScrollBar as ScrollBarEnum	A ScrollBar expression that indicates the scrollbar where the part is enabled or disabled.
Part as ScrollPartEnum	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 ScrollPartEnable property is automatically called, so the parts becomes enabled. Use the ScrollPartVisible property to add or remove buttons/parts in the control's scrollbar. Use the ScrollPartEnable property to specify enable or disable parts in the control's scrollbar. Use the ScrolPartCaption property to specify the caption of the scroll's part. Use the ScrollOrderParts property to customize the order of the buttons in the scroll bar.

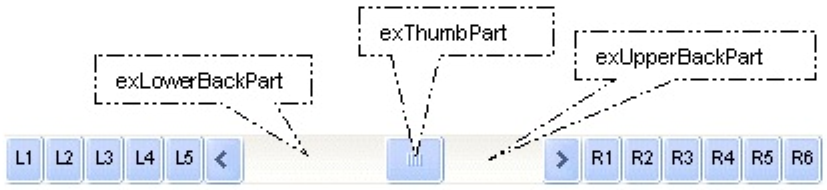


property CascadeFile.ScrollPartVisible(ScrollBar as ScrollBarEnum, Part as ScrollPartEnum) as Boolean

Indicates whether the specified scroll part is visible or hidden.

Type	Description
ScrollBar as ScrollBarEnum	A ScrollBar expression that indicates the scrollbar where the part is visible or hidden.
Part as ScrollPartEnum	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 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 ScrolPartCaption property to specify the caption of the scroll's part. Use the Background property to change the visual appearance for any part in the control's scroll bar. Use the ScrollOrderParts 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 CascadeFile1
    .BeginUpdate
        .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 AxCascadeFile1
    .BeginUpdate()
    .set_ScrollPartVisible(EXEXMILLERLib.ScrollBarEnum.exVScroll,
EXEXMILLERLib.ScrollPartEnum.exLeftB1Part Or
EXEXMILLERLib.ScrollPartEnum.exRightB1Part, True)
    .set_ScrollPartCaption(EXEXMILLERLib.ScrollBarEnum.exVScroll,
EXEXMILLERLib.ScrollPartEnum.exLeftB1Part, "<img></img>1")
    .set_ScrollPartCaption(EXEXMILLERLib.ScrollBarEnum.exVScroll,
EXEXMILLERLib.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 :

```
axCascadeFile1.BeginUpdate();
axCascadeFile1.set_ScrollPartVisible(EXEXMILLERLib.ScrollBarEnum.exVScroll,
EXEXMILLERLib.ScrollPartEnum.exLeftB1Part |
EXEXMILLERLib.ScrollPartEnum.exRightB1Part, true);
axCascadeFile1.set_ScrollPartCaption(EXEXMILLERLib.ScrollBarEnum.exVScroll,
EXEXMILLERLib.ScrollPartEnum.exLeftB1Part , "<img></img>1");
axCascadeFile1.set_ScrollPartCaption(EXEXMILLERLib.ScrollBarEnum.exVScroll,
EXEXMILLERLib.ScrollPartEnum.exRightB1Part, "<img></img>2");
axCascadeFile1.EndUpdate();
```

The following C++ sample adds up and down additional buttons to the control's vertical scroll bar :

```
m_cascadefile.BeginUpdate();
m_cascadefile.SetScrollPartVisible( 0 /*exVScroll*/, 32768  /*exLeftB1Part*/ | 32
/*exRightB1Part*/, TRUE );
```

```
m_cascadefile.SetScrollPartCaption( 0 /*exVScroll*/, 32768  /*exLeftB1Part*/ , _T("<img>
</img>1") );
m_cascadefile.SetScrollPartCaption( 0 /*exVScroll*/, 32 /*exRightB1Part*/ , _T("<img>
</img>2") );
m_cascadefile.EndUpdate();
```

The following VFP sample adds up and down additional buttons to the control's vertical scroll bar :

```
With thisform.CascadeFile1
    .BeginUpdate
        .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 CascadeFile.ScrollThumbSize(ScrollBar as ScrollBarEnum) as Long

Specifies the size of the thumb in the scrollbar.

Type	Description
ScrollBar as ScrollBarEnum	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 buttons in the horizontal scroll bar. Use the ScrollHeight property to specify the height of the horizontal scroll bar. Use the ScrollButtonHeight property to specify the height of the buttons in the vertical scroll bar. Use the ScrollPartVisible property to specify the visible parts in the control's scroll bar.

property CascadeFile.ScrollToolTip(ScrollBar as ScrollBarEnum) as String

Specifies the tooltip being shown when the user moves the scroll box.

Type	Description
ScrollBar as ScrollBarEnum	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 SortPartVisible property to specify the parts being visible in the control's scroll bar.

The following VB sample displays a tooltip when the user clicks and moves the thumb in the control's scroll bar:

```
Private Sub CascadeFile1_OffsetChanged(ByVal Horizontal As Boolean, ByVal NewVal As
Long)
    If (Not Horizontal) Then
        CascadeFile1.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 AxCascadeFile1_OffsetChanged(ByVal sender As System.Object, ByVal e As
AxEXEXMILLERLib._ICascadeFileEvents_OffsetChangedEvent) Handles
AxCascadeFile1.OffsetChanged
    If (Not e.horizontal) Then
        AxCascadeFile1.set_ScrollToolTip(EXEXMILLERLib.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 OnOffsetChangedCascadeFile1(BOOL Horizontal, long NewVal)
{
    if ( !Horizontal )
    {
        CString strFormat;
        strFormat.Format( _T("%i"), NewVal );
        m_cascadefile.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 axCascadeFile1_OffsetChanged(object sender,
AxEXEXMILLERLib._ICascadeFileEvents_OffsetChangedEvent e)
{
    if ( !e.horizontal )
        axCascadeFile1.set_ScrollToolTip(EXEXMILLERLib.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.CascadeFile1.ScrollToolTip(0) = "Record " + ltrim(str(newval))
EndIf
```

property CascadeFile.ScrollWidth as Long

Specifies the width of the vertical scrollbar.

Type	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 ScrollButtonWidth property to specify the width of the buttons in the horizontal scroll bar. Use the ScrollHeight property to specify the height of the horizontal scroll bar. Use the ScrollButtonHeight property to specify the height of the buttons in the vertical scroll bar. Use the ScrollPartVisible property to specify the visible parts in the control's scroll bar. Use the ScrollThumbSize property to define a fixed size for the scrollbar's thumb.

property CascadeFile.SelBackColor as Color

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

Type	Description
Color	A color expression that indicates the background color for selected items. 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 SelBackColor property specifies the background color for selected items. Use the SelForeColor property to specify the foreground color for selected items. Use the BackColor property to specify the control's background color. Use the ForeColor property to specify the control's foreground color.

property CascadeFile.Select as String

Selects a folder, giving its displaying name, relative or absolute path.

Type	Description
String	A String expression that specifies the file / folder to be selected. The parameter may include multiple files to be selected if separated by \r\n or | character.

The Select method selects a folder, giving its displaying name, relative or absolute path. The SelectMode property indicates the mode the control displays the last visible cascade column.

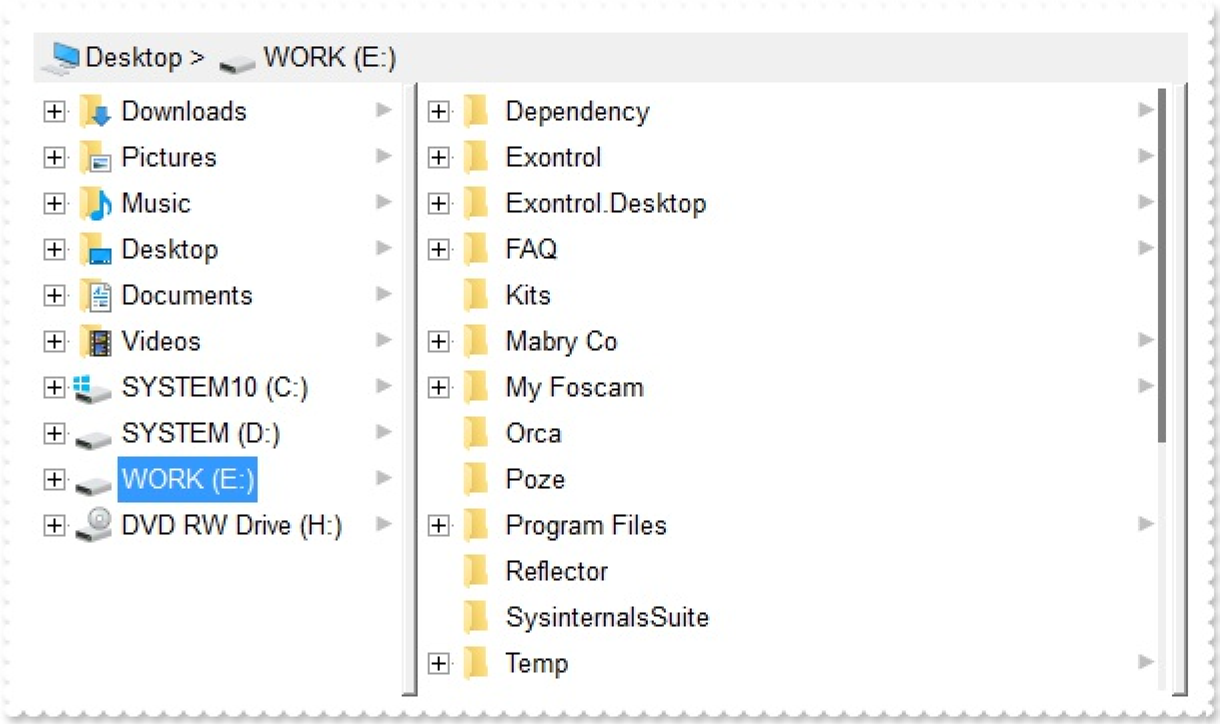
property CascadeFile.SelectMode as SelectModeEnum

Indicates the mode the control displays the last visible cascade column.

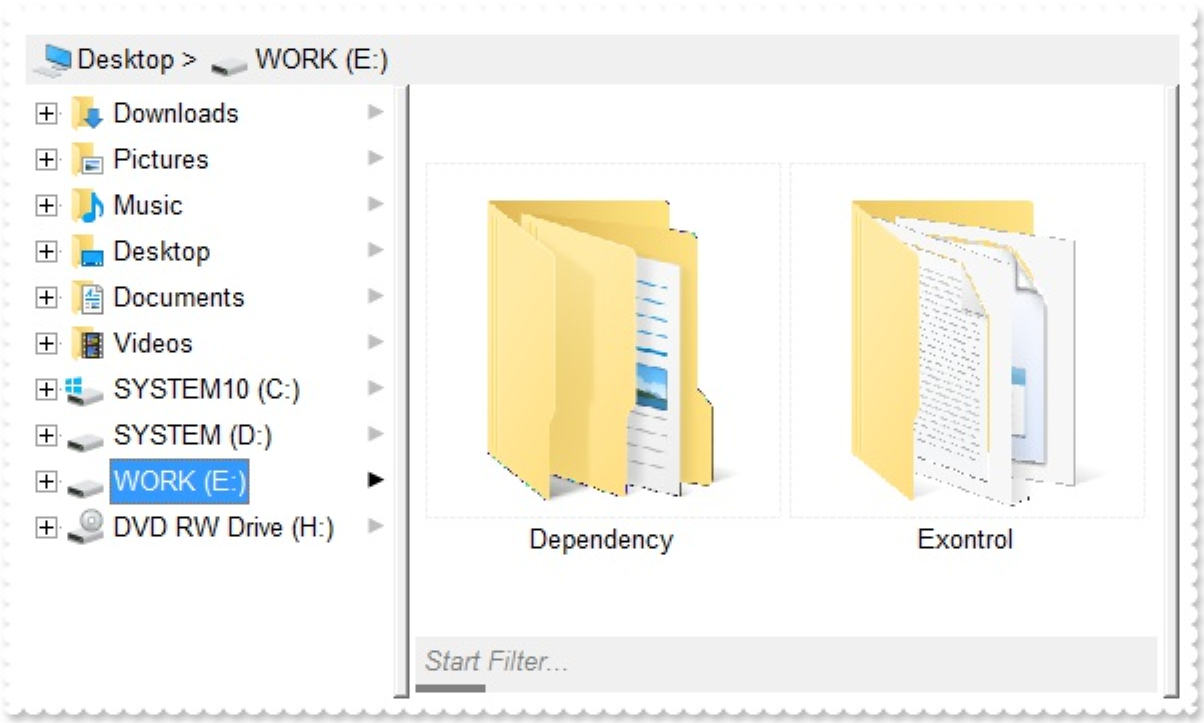
Type	Description
SelectModeEnum	A SelectModeEnum expression that specifies the mode the control displays the last visible cascade column.

The SelectMode property indicates the mode the control displays the last visible cascade column. The Select method selects a folder, giving its displaying name, relative or absolute path. The exDisableThumbnails flag of Mode property prevents any thumbnail mode.

The following screen shot shows the control while the SelectMode property is exSelectModeList:



The following screen shot shows the control while the SelectMode property is exSelectModeThumbnail:


Desktop > WORK (E:)
Downloads
Pictures
Music
Desktop
Documents
Videos
SYSTEM10 (C:)
SYSTEM (D:)
WORK (E:)
DVD RW Drive (H:)
Dependency
Exontrol
Start Filter...

property CascadeFile.SelForeColor as Color

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

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

The SelForeColor property specifies the foreground color for selected items. Use the SelBackColor property to specify the background color for selected items. Use the ForeColor property to specify the control's foreground color. Use the BackColor property to specify the control's background color.

property CascadeFile.ShowContextMenu as String

Specifies the object's context menu.

Type	Description
String	A String expression that specifies the commands to be displayed in the object's context menu. The ShowContextMenu property supports expressions, so you can combine the default context menu, with your own context menu for any file/folder.

By default, the ShowContextMenu property is empty. The ShowContextMenu property can be used to disable, update, remove or add new items. The ShowContextMenu property indicates the items to be displayed on the object's context menu. The AllowContextMenu property specifies whether the control shows the object's context menu when the user presses the right click over a file or folder.

For instance:

- "`[debug]` + menu" displays all item's identifiers in the control's default menu
- "menu replace `&Delete` with ``" removes the Delete command from any context menu
- "Popup(Item 1[id=1001],Item 2[id=1002],Item 3[id=1003]),[sep],Exit[def][id=1000]" defines a popup, a separator and a default item. This context menu will be shown any time, no mater if none, one or more files are selected
- "filecount > 1 ? `multiple selection[dis]` : menu" displays "multiple selection" displays the default context menu if the user selected a single file, else invokes the context menu for multiple-items selection
- "`Popup(Item 1[id=1001],Item 2[id=1002],Item 3[id=1003]),` + menu + `,Exit[id=1000]`" combine's the default selection context menu, so Popup is displayed at the top of the context menu, and the Exit item at the bottom. The Popup and Exit are always displayed, while the control's selection default context menu are shown only if available.
- "filecount = 0 ? `Popup(Item 1[id=1001],Item 2[id=1002],Item 3[id=1003]),[sep],Exit[def][id=1000]` : menu" defines a separated context menu when no file/folder is selected (control's background context menu). The default context menu is shown if the user right-clicks a file, folder or the selection.

The ShowContextMenu property supports the following predefined keywords:

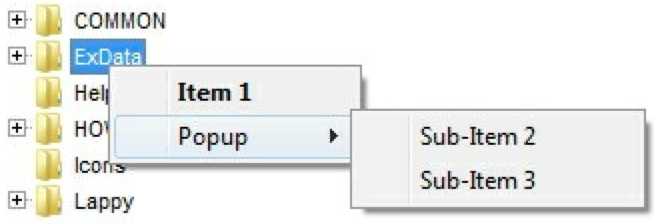
- **menu** keyword returns a string expression that defines the shell context menu's tostring representation
- **filecount** keywords returns a numeric expression that specifies the number of items/files/folders selected in the control
- **fileattr** keyword returns a numeric expression that specifies the attributes of the single-selected item in the control (the keyword's value is valid while the filecount property is

1)
- **filename** keyword returns a string expression that specifies the name of the single-selected item in the control (the keyword's value is valid while the filecount property is 1)
- **fileparsename** keyword returns a string expression that specifies the parsed name of the single-selected item in the control (the keyword's value is valid while the filecount property is 1)
- **filefullname** keyword returns a string expression that specifies the full name of the single-selected item in the control (the keyword's value is valid while the filecount property is 1)

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

The ShowContextMenu property indicates the list of commands to be displayed in the context menu, separated by comma (,). Each command must have an id parameter, that specifies the identifier of the command. Optional parameters are def for default item, and dis for disabled items. The sep parameter indicates a separator item. If adding new items to the object's context menu, use the ExecuteContextMenu property to get the identifier of the command to be executed during the ViewEndChanging event, when the Operation parameter is exExecuteContextMenu(21).

For instance, the ShowContextMenu property on *"Item 1[id=1][def],Popup[id=2](Sub-Item 2[id=2],[sep],Sub-Item 3[id=3])"* shows the context menu as following:



property CascadeFile.ShowFocusRect as Boolean

Retrieves or sets a value indicating whether the control draws a thin rectangle around the focused item.

Type	Description
Boolean	A boolean expression that specifies whether the focusing file or folder shows a dotted rectangle around.

By default, the ShowFocusRect property is True. For instance, you can use the ShowFocusRect property on False, if the SingleSel property is True, or using EBN to show the control's selection. Use the SelBackColor property to specify the background color for selected files or folders.

property CascadeFile.ShowImageList as Boolean

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

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

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

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

Shows the specified tooltip at given position.

Type	Description
ToolTip as String	The ToolTip parameter can be any of the following: <ul><li>NULL(BSTR) or "<null>"(string) to indicate that the tooltip for the object being hovered is not changed</li><li>A String expression that indicates the description of the tooltip, that supports built-in HTML format (adds, replaces or changes the object's tooltip)</li></ul>
Title as Variant	The Title parameter can be any of the following: <ul><li>missing (VT_EMPTY, VT_ERROR type) or "<null>" (string) the title for the object being hovered is not changed.</li><li>A String expression that indicates the title of the tooltip (no built-in HTML format) (adds, replaces or changes the object's title)</li></ul>
Alignment as Variant	A long expression that indicates the alignment of the tooltip relative to the position of the cursor. If missing (VT_EMPTY, VT_ERROR) the alignment of the tooltip for the object being hovered is not changed. The Alignment parameter can be one of the following: <ul><li>0 - exTopLeft</li><li>1 - exTopRight</li><li>2 - exBottomLeft</li><li>3 - exBottomRight</li><li>0x10 - exCenter</li><li>0x11 - exCenterLeft</li><li>0x12 - exCenterRight</li><li>0x13 - exCenterTop</li><li>0x14 - exCenterBottom</li></ul> By default, the tooltip is aligned relative to the top-left corner (0 - exTopLeft).

X as Variant	Specifies the horizontal position to display the tooltip as one of the following: • missing (VT_EMPTY, VT_ERROR type), indicates that the tooltip is shown on its default position / current cursor position (ignored) • -1, indicates the current horizontal position of the cursor (current x-position) • a numeric expression that indicates the horizontal screen position to show the tooltip (fixed screen x-position) • a string expression that indicates the horizontal displacement relative to default position to show the tooltip (moved)
Y as Variant	Specifies the vertical position to display the tooltip as one of the following: • missing (VT_EMPTY, VT_ERROR type), indicates that the tooltip is shown on its default position / current cursor position (ignored) • -1, indicates the current vertical position of the cursor (current y-position) • a numeric expression that indicates the vertical screen position to show the tooltip (fixed screen y-position) • a string expression that indicates the vertical displacement relative to default position to show the tooltip (displacement)

Use the ShowToolTip method to display a custom tooltip at specified position or to update the object's tooltip, title or position. You can call the ShowToolTip method during the MouseMove event. Use the ToolTipPopDelay property specifies the period in ms of time the ToolTip remains visible if the mouse pointer is stationary within a control. The ToolTipDelay property specifies the time in ms that passes before the ToolTip appears. Use the ToolTipWidth property to specify the width of the tooltip window. Use the ToolTipFont property to change the tooltip's font. Use the Background(exToolTipAppearance) property indicates the visual appearance of the borders of the tooltips. Use the Background(exToolTipBackColor) property indicates the tooltip's background color. Use the Background(exToolTipForeColor) property indicates the tooltip's foreground color.

For instance:

- ShowToolTip(`<null>`,`<null>`,,`+8`,`+8`), shows the tooltip of the object moved relative to its default position

- ShowToolTip(`<null>`,`new title`), adds, changes or replaces the title of the object's tooltip
- ShowToolTip(`new content`), adds, changes or replaces the object's tooltip
- ShowToolTip(`new content`,`new title`), shows the tooltip and title at current position
- ShowToolTip(`new content`,`new title`,,`+8`,`+8`), shows the tooltip and title moved relative to the current position
- ShowToolTip(`new content`,``,,128,128), displays the tooltip at a fixed position
- ShowToolTip(``,``), hides the tooltip

The ToolTip parameter supports the built-in HTML format like follows:

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

- **<r>** right aligns the text
- **<c>** centers the text
- **<br>** forces a line-break
- **<img>number[:width]</img>** inserts an icon inside the text. The number indicates the index of the icon being inserted. Use the Images method to assign a list of icons to your chart. The last 7 bits in the high significant byte of the number expression indicates the identifier of the skin being used to paint the object. Use the Add method to add new skins to the control. If you need to remove the skin appearance from a part of the control you need to reset the last 7 bits in the high significant byte of the color being applied to the part. The width is optional and indicates the width of the icon being inserted. Using the width option you can overwrite multiple icons getting a nice effect. By default, if the width field is missing, the width is 18 pixels.
- **<img>key[:width]</img>** inserts a custom size picture into the text being previously loaded using the HTMLPicture property. The Key parameter indicates the key of the picture being displayed. The Width parameter indicates a custom size, if you require to stretch the picture, else the original size of the picture is used.
- **&** glyph characters as **&amp;** (&), **&lt;** (<), **&gt;** (>), **&qout;** (") and **&#number;** (the character with specified code), For instance, the &#8364; displays the EUR character. The **&** ampersand is only recognized as markup when it is followed by a known letter or a #character and a digit. For instance if you want to display <b>bold</b> in HTML caption you can use &lt;b&gt;bold&lt;/b&gt;
- **<off offset> ... </off>** defines the vertical offset to display the text/element. The offset parameter defines the offset to display the element. This tag is inheritable, so the offset is keep while the associated </off> tag is found. You can use the <off offset> HTML tag in combination with the <font face;size> to define a smaller or a larger font to be displayed. For instance: "Text with <font ;7><**off** 6>subscript" displays the text such as: Text with $_{\text{subscript}}$ The "Text with <font ;7><**off** -6>superscript" displays the text such as: Text with $^{\text{subscript}}$
- **<gra rrggbb;mode;blend> ... </gra>** defines a gradient text. The text color or <fgcolor> defines the starting gradient color, while the rr/gg/bb represents the red/green/blue values of the ending color, 808080 if missing as gray. The mode is a value between 0 and 4, 1 if missing, and blend could be 0 or 1, 0 if missing. The <font> HTML tag can be used to define the height of the font. Any of the rrggbb, mode or blend field may not be specified. The <gra> with no fields, shows a vertical gradient color from the current text color to gray (808080). For instance the "<font ;18><**gra** FFFFFF;1;1>gradient-center</**gra**></font>" generates the following picture:

gradient-center

- **<out rrggbb;width> ... </out>** shows the text with outlined characters, where rr/gg/bb represents the red/green/blue values of the outline color, 808080 if missing as gray, width indicates the size of the outline, 1 if missing. The text color or <fgcolor> defines the color to show the inside text. The <font> HTML tag can be used to define the height of the font. For instance the "<font ;31><**out** 000000><fgcolor=FFFFFF>outlined</fgcolor></**out**></font>" generates the following picture:

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

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

outline anti-aliasing

property CascadeFile.SingleSel as Boolean

Retrieves or sets a value indicating whether control support single or multiples selection.

Type	Description
Boolean	A boolean expression indicating whether control support single or multiples selection.

By default, the SingleSel property is True. The SingleSel property specifies whether the user can select single or multiple items in the control. If the SingleSel property is True, the control may select only a single file/folder. If the SingleSel is False, the user can select multiple files/folders by dragging a box, or by using SHIFT and CTRL keys. Use the SelForeColor property to specify the foreground color for selected files or folders. Use the SelBackColor property to specify the background color for selected files or folders. For instance, you can use the ShowFocusRect property on False, if the SingleSel property is True, or using EBN to show the control's selection. The FullRowSelect property enables full-row selection in the control.

property CascadeFile.SplitViewHeight as String

Specifies the height of split panels, separated by comma.

Type	Description
String	A String expression that specifies the height for each vertical split-panel, separated by comma character.

By default, the SplitViewHeight property is "", so no additional view is displayed. The SplitViewHeight property specifies the height of split panels, separated by comma. The AllowSplitView property specifies whether the user can split the control into multiple-views. The Background(exHSplitBar) property specifies the visual appearance of the control's split bar (horizontal split bar)

property CascadeFile.StatusBarLabel as String

Specifies the HTML label the control's status bar is displaying.

Type	Description
String	A String expression that specifies the HTML label the control's status bar is displaying.

By default, the StatusBarLabel property is "<%1%> >". The StatusBarLabel property specifies the HTML label the control's status bar is displaying. The StatusBarVisible property specifies whether the control's status bar is visible or hidden. The HeaderVisible property retrieves or sets a value that indicates whether the control's header bar is visible or hidden. Use the ColumnsVisible property to hide or show a column. The Background(exStatusBackColor) / Background(exStatusForeColor) specifies the status bar's background / foreground color. The Background(exStatusPanelBackColor) specifies the status panel's background color.

Inside the StatusBarLabel property the following:

- **<%0%>** indicates the full path of the browsed folder aka C:\Program Files\Exontrol
- **<%1%>** indicates the name of the browsed folder aka SYSTEM (C:) instead of C:\, or Temp for C:\Temp
- **<%2%>** indicates the parsed name of the browsed folder aka C:\ for SYSTEM (C:) or Temp for C:\Temp
- **<%3%>** indicates the name of the browsed folder relative to the folder being specified by the ExplorerFromHere property. If the ExplorerFromHere property is not specified, the <%3%> is identical with <%0%>. If the ExplorerFromHere property refers a folder its name is not shown on <%3%>, so only relative name is displayed.

The StatusBarLabel property supports the built-in HTML format like follows:

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

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

- **<fgcolor rrggbb> ... </fgcolor>** or <fgcolor=rrggbb> ... </fgcolor> displays text with a specified foreground color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<bgcolor rrggbb> ... </bgcolor>** or <bgcolor=rrggbb> ... </bgcolor> displays text with a specified background color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<solidline rrggbb> ... </solidline>** or <solidline=rrggbb> ... </solidline> draws a solid-line on the bottom side of the current text-line, of specified RGB color. The <solidline> ... </solidline> draws a black solid-line on the bottom side of the current text-line. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<dotline rrggbb> ... </dotline>** or <dotline=rrggbb> ... </dotline> draws a dot-line on the bottom side of the current text-line, of specified RGB color. The <dotline> ... </dotline> draws a black dot-line on the bottom side of the current text-line. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<upline> ... </upline>** draws the line on the top side of the current text-line (requires <solidline> or <dotline>).
- **<r>** right aligns the text
- **<c>** centers the text
- **<br>** forces a line-break
- **<img>number[:width]</img>** inserts an icon inside the text. The number indicates the index of the icon being inserted. Use the Images method to assign a list of icons to your chart. The last 7 bits in the high significant byte of the number expression indicates the identifier of the skin being used to paint the object. Use the Add method to add new skins to the control. If you need to remove the skin appearance from a part of the control you need to reset the last 7 bits in the high significant byte of the color being applied to the part. The width is optional and indicates the width of the icon being inserted. Using the width option you can overwrite multiple icons getting a nice effect. By default, if the width field is missing, the width is 18 pixels.
- **<img>key[:width]</img>** inserts a custom size picture into the text being previously loaded using the HTMLPicture property. The Key parameter indicates the key of the picture being displayed. The Width parameter indicates a custom size, if you require to stretch the picture, else the original size of the picture is used.
- **&** glyph characters as **&amp;** (&), **&lt;** (<), **&gt;** (>), **&qout;** (") and **&#number;** (the character with specified code), For instance, the &#8364; displays the EUR character. The **&** ampersand is only recognized as markup when it is followed by a known letter or a #character and a digit. For instance if you want to display <b>bold</b> in HTML caption you can use &lt;b&gt;bold&lt;/b&gt;
- **<off offset> ... </off>** defines the vertical offset to display the text/element. The offset parameter defines the offset to display the element. This tag is inheritable, so the

offset is keep while the associated </off> tag is found. You can use the <off offset> HTML tag in combination with the <font face;size> to define a smaller or a larger font to be displayed. For instance: "Text with <font ;7><**off** 6>subscript" displays the text such as: Text with $_{\text{subscript}}$ The "Text with <font ;7><**off** -6>superscript" displays the text such as: Text with $^{\text{subscript}}$

- **<gra rrggbb;mode;blend> ... </gra>** defines a gradient text. The text color or <fgcolor> defines the starting gradient color, while the rr/gg/bb represents the red/green/blue values of the ending color, 808080 if missing as gray. The mode is a value between 0 and 4, 1 if missing, and blend could be 0 or 1, 0 if missing. The <font> HTML tag can be used to define the height of the font. Any of the rrggbb, mode or blend field may not be specified. The <gra> with no fields, shows a vertical gradient color from the current text color to gray (808080). For instance the "<font ;18><**gra** FFFFFF;1;1>gradient-center</**gra**></font>" generates the following picture:

gradient-center

- **<out rrggbb;width> ... </out>** shows the text with outlined characters, where rr/gg/bb represents the red/green/blue values of the outline color, 808080 if missing as gray, width indicates the size of the outline, 1 if missing. The text color or <fgcolor> defines the color to show the inside text. The <font> HTML tag can be used to define the height of the font. For instance the "<font ;31><**out** 000000><fgcolor=FFFFFF>outlined</fgcolor></**out**></font>" generates the following picture:

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

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

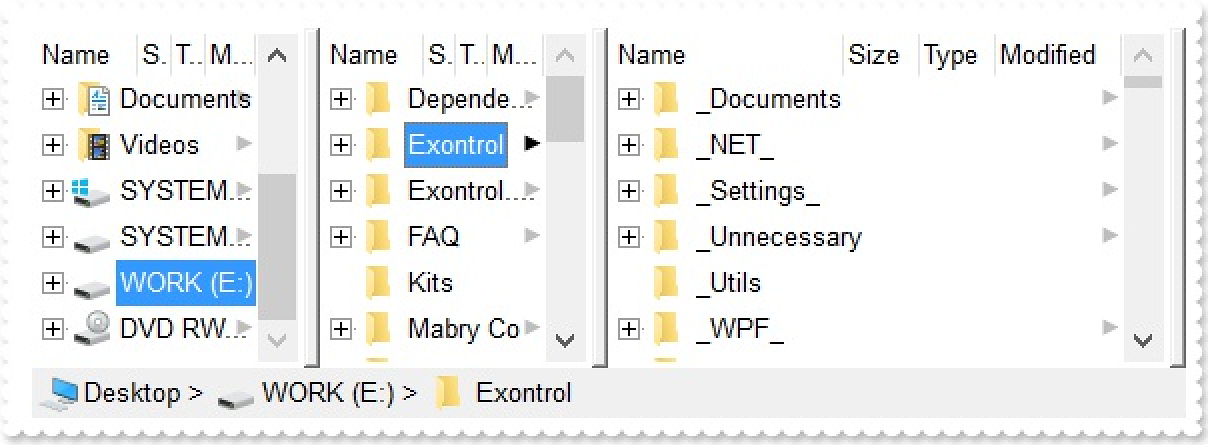
outline anti-aliasing

property CascadeFile.StatusBarVisible as StatusBarAnchorEnum

Specifies whether the control's status bar is visible or hidden.

Type	Description
StatusBarAnchorEnum	A StatusBarAnchorEnum expression that specifies whether the control's status bar is visible or hidden.

By default, the StatusBarVisible property is exStatusBarAnchorTop, and so the status bar is visible and aligned to the top side of the control. The StatusBarVisible property specifies whether the control's status bar is visible or hidden. The StatusBarLabel property specifies the HTML label the control's status bar is displaying. The HeaderVisible property retrieves or sets a value that indicates whether the control's header bar is visible or hidden. Use the ColumnsVisible property to hide or show a column. The Background(exStatusBackColor) / Background(exStatusForeColor) specifies the status bar's background / foreground color. The Background(exStatusPanelBackColor) specifies the status panel's background color.



property CascadeFile.Template as String

Specifies the control's template.

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

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

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

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

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

The Template script is composed by lines of instructions. Instructions are separated by "\n\r" (newline) characters.

An instruction can be one of the following:

- **Dim** list of variables *Declares the variables. Multiple variables are separated by commas. (Sample: Dim h, h1, h2)*
- variable **=** property(list of arguments) *Assigns the result of the property to a variable. The "variable" is the name of a declared variable. The "property" is the property name of the object in the context. The "list or arguments" may include variables or values*

separated by commas. (Sample: h = InsertItem(0,"New Child"))

- property**(** list of arguments **)** = value *Changes the property. The value can be a variable, a string, a number, a boolean value or a RGB value.*
- method**(** list of arguments **)** *Invokes the method. The "list or arguments" may include variables or values separated by commas.*
- **{** *Beginning the object's context. The properties or methods called between { and } are related to the last object returned by the property prior to { declaration.*
- **}** *Ending the object's context*
- object**.** property(list of arguments)**.**property(list of arguments).... *The .(dot) character splits the object from its property. For instance, the Columns.Add("Column1").HeaderBackColor = RGB(255,0,0), adds a new column and changes the column's header back color.*

The Template supports the following general functions:

- **RGB(**R,G,B**)** *property retrieves an RGB value, where the R, G, B are byte values that indicates the R G B values for the color being specified. For instance, the following code changes the control's background color to red: BackColor = RGB(255,0,0)*
- **CreateObject(**progID**)** *property creates and retrieves a single uninitialized object of the class associated with a specified program identifier*

property CascadeFile.TemplateDef as Variant

Defines inside variables for the next Template/ExecuteTemplate call.

Type	Description
Variant	A string expression that indicates the Dim declaration, or any Object expression to be assigned to previously declared variables.

The TemplateDef property has been added to allow programming languages such as dBASE Plus to set control's properties with multiple parameters. It is known that programming languages such as **dBASE Plus or XBasic from AlphaFive**, does not support setting a property with multiple parameters. In other words, these programming languages does not support something like *Property(Parameters)* = *Value*, so our controls provide an alternative using the TemplateDef method. The first call of the TemplateDef should be a declaration such as "Dim a,b" which means the next 2 calls of the TemplateDef defines the variables a and b. The next call should be Template or ExecuteTemplate property which can use the variable a and b being defined previously.

So, calling the TemplateDef property should be as follows:

```
with (Control)
   TemplateDef = [Dim var_Column]
   TemplateDef = var_Column
   Template = [var_Column.Def(4) = 255]
endwith
```

This sample allocates a variable var_Column, assigns the value to the variable (the second call of the TemplateDef), and the Template call uses the var_Column variable (as an object), to call its Def property with the parameter 4.

Let's say we need to define the background color for a specified column, so we need to call the Def(exCellBackColor) property of the column, to define the color for all cells in the column.

The following **VB6** sample shows setting the Def property such as:

```
With Control
   .Columns.Add("Column 1").Def(exCellBackColor) = 255
   .Columns.Add "Column 2"
   .Items.AddItem 0
   .Items.AddItem 1
```

```
    .Items.AddItem 2
End With
```

In **dBASE Plus**, calling the Def(4) has no effect, instead using the TemplateDef helps you to use properly the Def property as follows:

```
local Control,var_Column

Control = form.Activex1.nativeObject
// Control.Columns.Add("Column 1").Def(4) = 255
var_Column = Control.Columns.Add("Column 1")
with (Control)
    TemplateDef = [Dim var_Column]
    TemplateDef = var_Column
    Template = [var_Column.Def(4) = 255]
endwith
Control.Columns.Add("Column 2")
Control.Items.AddItem(0)
Control.Items.AddItem(1)
Control.Items.AddItem(2)
```

The equivalent sample for **XBasic in A5**, is as follows:

```
Dim Control as P
Dim var_Column as P

Control = topparent:CONTROL_ACTIVEX1.activex
' Control.Columns.Add("Column 1").Def(4) = 255
var_Column = Control.Columns.Add("Column 1")
Control.TemplateDef = "Dim var_Column"
Control.TemplateDef = var_Column
Control.Template = "var_Column.Def(4) = 255"

Control.Columns.Add("Column 2")
Control.Items.AddItem(0)
Control.Items.AddItem(1)
Control.Items.AddItem(2)
```

The samples just call the Column.Def(4) = Value, using the TemplateDef. The first call of TemplateDef property is "Dim var_Column", which indicates that the next call of the TemplateDef will defines the value of the variable var_Column, in other words, it defines the object var_Column. The last call of the Template property uses the var_Column member to use the x-script and so to set the Def property so a new color is being assigned to the column.

The TemplateDef, Template and ExecuteTemplate support x-script language (Template script of the Exontrols), like explained bellow:

The Template or x-script is composed by lines of instructions. Instructions are separated by "\n\r" (newline characters) or ";" character. The ; character may be available only for newer versions of the components.

An x-script instruction/line can be one of the following:

- **Dim** list of variables *Declares the variables. Multiple variables are separated by commas. (Sample: Dim h, h1, h2)*
- variable **=** property(list of arguments) *Assigns the result of the property to a variable. The "variable" is the name of a declared variable. The "property" is the property name of the object in the context. The "list or arguments" may include variables or values separated by commas. (Sample: h = InsertItem(0,"New Child"))*
- property**(** list of arguments **)** = value *Changes the property. The value can be a variable, a string, a number, a boolean value or a RGB value.*
- method**(** list of arguments **)** *Invokes the method. The "list or arguments" may include variables or values separated by commas.*
- **{** *Beginning the object's context. The properties or methods called between { and } are related to the last object returned by the property prior to { declaration.*
- **}** *Ending the object's context*
- object**.** property(list of arguments)**.**property(list of arguments).... *The .(dot) character splits the object from its property. For instance, the Columns.Add("Column1").HeaderBackColor = RGB(255,0,0), adds a new column and changes the column's header back color.*

The x-script may uses constant expressions as follow:

- *boolean* expression with possible values as *True* or *False*
- *numeric* expression may starts with 0x which indicates a hexa decimal representation, else it should starts with digit, or +/- followed by a digit, and . is the decimal separator. *Sample: 13 indicates the integer 13, or 12.45 indicates the double expression 12,45*
- *date* expression is delimited by **#** character in the format #mm/dd/yyyy hh:mm:ss#. *Sample: #31/12/1971# indicates the December 31, 1971*
- *string* expression is delimited by **"** or **`** characters. If using the ` character, please

make sure that it is different than ' which allows adding comments inline. *Sample: "text" indicates the string text.*

Also , the template or x-script code may support general functions as follows:

- **Me** *property indicates the original object.*
- **RGB(**R,G,B**)** *property retrieves an RGB value, where the R, G, B are byte values that indicates the R G B values for the color being specified. For instance, the following code changes the control's background color to red: BackColor = RGB(255,0,0)*
- **LoadPicture(**file**)** *property loads a picture from a file or from BASE64 encoded strings, and returns a Picture object required by the picture properties.*
- **CreateObject(**progID**)** *property creates and retrieves a single uninitialized object of the class associated with a specified program identifier.*

method CascadeFile.TemplatePut (newVal as Variant)

Defines inside variables for the next Template/ExecuteTemplate call.

Type	Description
newVal as Variant	A string expression that indicates the Dim declaration, or any Object expression to be assigned to previously declared variables.

The TemplatePut method / TemplateDef property has been added to allow programming languages such as dBASE Plus to set control's properties with multiple parameters. It is known that programming languages such as **dBASE Plus or XBasic from AlphaFive**, does not support setting a property with multiple parameters. In other words, these programming languages does not support something like *Property(Parameters) = Value*, so our controls provide an alternative using the TemplateDef / TemplatePut method. The first call of the TemplateDef should be a declaration such as "Dim a,b" which means the next 2 calls of the TemplateDef defines the variables a and b. The next call should be Template or ExecuteTemplate property which can use the variable a and b being defined previously.

The TemplateDef, TemplatePut, Template and ExecuteTemplate support x-script language (Template script of the Exontrols), like explained bellow:

The Template or x-script is composed by lines of instructions. Instructions are separated by "\n\r" (newline characters) or ";" character. The ; character may be available only for newer versions of the components.

An x-script instruction/line can be one of the following:

- **Dim** list of variables *Declares the variables. Multiple variables are separated by commas. (Sample: Dim h, h1, h2)*
- variable **=** property(list of arguments) *Assigns the result of the property to a variable. The "variable" is the name of a declared variable. The "property" is the property name of the object in the context. The "list or arguments" may include variables or values separated by commas. (Sample: h = InsertItem(0,"New Child"))*
- property**(** list of arguments **)** = value *Changes the property. The value can be a variable, a string, a number, a boolean value or a RGB value.*
- method**(** list of arguments **)** *Invokes the method. The "list or arguments" may include variables or values separated by commas.*
- **{** *Beginning the object's context. The properties or methods called between { and } are related to the last object returned by the property prior to { declaration.*
- **}** *Ending the object's context*
- object**.** property(list of arguments)**.**property(list of arguments).... *The .(dot) character splits the object from its property. For instance, the*

Columns.Add("Column1").HeaderBackColor = RGB(255,0,0), adds a new column and changes the column's header back color.

The x-script may uses constant expressions as follow:

- *boolean* expression with possible values as *True* or *False*
- *numeric* expression may starts with 0x which indicates a hexa decimal representation, else it should starts with digit, or +/- followed by a digit, and . is the decimal separator. *Sample: 13 indicates the integer 13, or 12.45 indicates the double expression 12,45*
- *date* expression is delimited by **#** character in the format #mm/dd/yyyy hh:mm:ss#. *Sample: #31/12/1971# indicates the December 31, 1971*
- *string* expression is delimited by **"** or **`** characters. If using the ` character, please make sure that it is different than ' which allows adding comments inline. *Sample: "text" indicates the string text.*

Also , the template or x-script code may support general functions as follows:

- **Me** *property indicates the original object.*
- **RGB(**R,G,B**)** *property retrieves an RGB value, where the R, G, B are byte values that indicates the R G B values for the color being specified. For instance, the following code changes the control's background color to red: BackColor = RGB(255,0,0)*
- **LoadPicture(**file**)** *property loads a picture from a file or from BASE64 encoded strings, and returns a Picture object required by the picture properties.*
- **CreateObject(**progID**)** *property creates and retrieves a single uninitialized object of the class associated with a specified program identifier.*

property CascadeFile.ToolTipDelay as Long

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

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

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

property CascadeFile.ToolTipFont as IFontDisp

Retrieves or sets the tooltip's font.

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

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

property CascadeFile.ToolTipPopDelay as Long

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

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

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

property CascadeFile.ToolTipWidth as Long

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

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

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

property CascadeFile.UseTabKey as Boolean

Retrieves or sets a value that specifies whether the Tab or SHIFT + Tab key navigates through the cascading columns.

Type	Description
Boolean	A Boolean expression that specifies whether Tab or SHIFT + Tab key navigates through the cascading columns.

By default, the UseTabKey property is True, which indicates that TAB activates the next cascade column, based on the current selection. The UseTabKey property specifies whether the Tab or SHIFT + Tab key navigates through the cascading columns.

property CascadeFile.Version as String

Retrieves the control's version.

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

The Version property is read-only. The Version property specifies the version of the control you are running.

property CascadeFile.VisualAppearance as Appearance

Retrieves the control's appearance.

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

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

ExMiller events

Tip The /COM object can be placed on a HTML page (with usage of the HTML object tag: <object classid="clsid:...">) using the class identifier: {A44EFDCD-D526-4E58-909D-193FD6519290}. The object's program identifier is: "Exontrol.ExMiller". The /COM object module is: "ExMiller.dll"

The component supports the following events:

Name	Description
AnchorClick	Occurs when an anchor element is clicked.
Click	Occurs when the user presses and then releases the left mouse button over the control.
DblClick	Occurs when the user dblclk the left mouse button over an object.
Event	Notifies the application once the control fires an event.
KeyDown	Occurs when the user presses a key while an object has the focus.
KeyPress	Occurs when the user presses and releases an ANSI key.
KeyUp	Occurs when the user releases a key while an object has the focus.
MouseDown	Occurs when the user presses a mouse button.
MouseMove	Occurs when the user moves the mouse.
MouseUp	Occurs when the user releases a mouse button.
RClick	Occurs once the user right clicks the control.
ViewEndChanging	Occurs once the user changed the view.
ViewStartChanging	Occurs once the user is about to change the view.

event AnchorClick (AnchorID as String, Options as String)

Occurs when an anchor element is clicked.

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

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

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

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

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

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

C#
```
private void AnchorClick(object sender,
AxEXMILLERLib._IGaugeEvents_AnchorClickEvent e)
{
}
```

C++

```
void OnAnchorClick(LPCTSTR   AnchorID,LPCTSTR   Options)
{
}
```

C++ Builder

```
void __fastcall AnchorClick(TObject *Sender,BSTR   AnchorID,BSTR   Options)
{
}
```

Delphi

```
procedure AnchorClick(ASender: TObject; AnchorID : WideString;Options :
WideString);
begin
end;
```

Delphi 8 (.NET only)

```
procedure AnchorClick(sender: System.Object; e:
AxEXMILLERLib._IGaugeEvents_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
AxEXMILLERLib._IGaugeEvents_AnchorClickEvent) Handles AnchorClick
End Sub
```

VB6

```
Private Sub AnchorClick(ByVal AnchorID As String,ByVal Options As String)
End Sub
```

VBA

```
Private Sub AnchorClick(ByVal AnchorID As String,ByVal Options As String)
End Sub
```

VFP

```
LPARAMETERS AnchorID,Options
```

Xbas...

```
PROCEDURE OnAnchorClick(oGauge,AnchorID,Options)
```

```
RETURN
```

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

Java…

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

VBSc…

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

Visual Data…

```
Procedure OnComAnchorClick String   llAnchorID String   llOptions
   Forward Send OnComAnchorClick llAnchorID llOptions
End_Procedure
```

Visual Objects

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

X++

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

XBasic

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

dBASE

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

event Click ()

Occurs when the user clicks the list.

Type	Description

Use the Click event to notify your application when the user clicks the list. Use the MouseDown or MouseUp event to notify your application when the user presses or releases the one of the mouse buttons.

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

C#
```
private void Click(object sender)
{
}
```

VB
```
Private Sub Click(ByVal sender As System.Object) Handles Click
End Sub
```

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

C#
```
private void ClickEvent(object sender, EventArgs e)
{
}
```

C++
```
void OnClick()
{
}
```

C++ Builder
```
void __fastcall Click(TObject *Sender)
{
}
```

Delphi
```
procedure Click(ASender: TObject; );
begin
end;
```

Delphi 8 (.NET only)
```
procedure ClickEvent(sender: System.Object; e: System.EventArgs);
begin
end;
```

Powe…
```
begin event Click()
end event Click
```

VB.NET
```
Private Sub ClickEvent(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles ClickEvent
End Sub
```

VB6
```
Private Sub Click()
End Sub
```

VBA
```
Private Sub Click()
End Sub
```

VFP
```
LPARAMETERS nop
```

Xbas…
```
PROCEDURE OnClick(oCascadeFile)
RETURN
```

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

Java…
```
<SCRIPT EVENT="Click()" LANGUAGE="JScript">
</SCRIPT>
```

VBSc…
```
<SCRIPT LANGUAGE="VBScript">
Function Click()
End Function
</SCRIPT>
```

Visual Data…
```
Procedure OnComClick
   Forward Send OnComClick
End_Procedure
```

Visual Objects
```
METHOD OCX_Click() CLASS MainDialog
RETURN NIL
```

X++
```
void onEvent_Click()
{
}
```

XBasic

```
function Click as v ()
end function
```

dBASE

```
function nativeObject_Click()
return
```

event DblClick (Shift as Integer, X as OLE_XPOS_PIXELS, Y as OLE_YPOS_PIXELS)

Fired when the user double clicks an item.

Type	Description
Shift as Integer	An integer that corresponds to the state of the SHIFT, CTRL, and ALT keys.
X as OLE_XPOS_PIXELS	A single that specifies the current X location of the mouse pointer. The x values is always expressed in container coordinates.
Y as OLE_YPOS_PIXELS	A single that specifies the current Y location of the mouse pointer. The y values is always expressed in container coordinates

The DblClk event is fired whenever the user double clicks a file or a folder. Use the ExecuteContextCommand method to invoke a command from the file's context menu.

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

C#
```
private void DblClick(object sender)
{
}
```

VB
```
Private Sub DblClick(ByVal sender As System.Object) Handles DblClick
End Sub
```

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

C#
```
private void DblClick(object sender, EventArgs e)
{
}
```

C++
```
void OnDblClick()
{
}
```

C++ Builder
```
void __fastcall DblClick(TObject *Sender)
{
}
```

Delphi

```
procedure DblClick(ASender: TObject; );
begin
end;
```

Delphi 8 (.NET only)

```
procedure DblClick(sender: System.Object; e: System.EventArgs);
begin
end;
```

Powe…

```
begin event DblClick()
end event DblClick
```

VB.NET

```
Private Sub DblClick(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles DblClick
End Sub
```

VB6

```
Private Sub DblClick()
End Sub
```

VBA

```
Private Sub DblClick()
End Sub
```

VFP

```
LPARAMETERS nop
```

Xbas…

```
PROCEDURE OnDblClick(oCascadeFile)
RETURN
```

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

Java…

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

VBSc…

```
<SCRIPT LANGUAGE="VBScript">
Function DblClick()
End Function
</SCRIPT>
```

Visual Data…

```
Procedure OnComDblClick
    Forward Send OnComDblClick
End_Procedure
```

Visual Objects

```
METHOD OCX_DblClick() CLASS MainDialog
RETURN NIL
```

X++

```
void onEvent_DblClick()
{
}
```

XBasic

```
function DblClick as v ()
end function
```

dBASE

```
function nativeObject_DblClick()
return
```

event Event (EventID as Long)

Notifies the application once the control fires an event.

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

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

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

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

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, AxEXMILLERLib._ICascadeFileEvents_EventEvent e)
{
}
```

C++
```
void OnEvent(long EventID)
{
}
```

C++ Builder

```
void __fastcall Event(TObject *Sender,long EventID)
{
}
```

Delphi

```
procedure Event(ASender: TObject; EventID : Integer);
begin
end;
```

Delphi 8 (.NET only)

```
procedure Event(sender: System.Object; e:
AxEXMILLERLib._ICascadeFileEvents_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
AxEXMILLERLib._ICascadeFileEvents_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(oCascadeFile,EventID)
RETURN
```

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

Java…

```
<SCRIPT EVENT="Event(EventID)" LANGUAGE="JScript">
</SCRIPT>
```

VBSc...

```
<SCRIPT LANGUAGE="VBScript">
Function Event(EventID)
End Function
</SCRIPT>
```

Visual Data...

```
Procedure OnComEvent Integer llEventID
   Forward Send OnComEvent llEventID
End_Procedure
```

Visual Objects

```
METHOD OCX_Event(EventID) CLASS MainDialog
RETURN NIL
```

X++

```
void onEvent_Event(int _EventID)
{
}
```

XBasic

```
function Event as v (EventID as N)
end function
```

dBASE

```
function nativeObject_Event(EventID)
return
```

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

Let's presume that we need to handle the BarParentChange event to change the _Cancel parameter from false to true, which fires the "Error executing code: FormActiveXControl (data source), method onEvent_BarParentChange called with invalid parameters." We need to know the identifier of the BarParentChange event (each event has an unique identifier and it is static, defined in the control's type library). If you are not familiar with what a type library means just handle the Event of the control as follows:

```
// Notifies the application once the control fires an event.
void onEvent_Event(int _EventID)
{
   print cascadefile1.EventParam(-2).toString();
}
```

This code allows you to display the information for each event of the control being fired as in the list bellow:

```
"MouseMove/-606( 1 , 0 , 145 , 36 )" VT_BSTR
"BarParentChange/125( 192998632 , 'B' , 192999592 , =false )" VT_BSTR
"BeforeDrawPart/54( 2 , -1962866148 , =0 , =0 , =0 , =0 , =false )" VT_BSTR
"AfterDrawPart/55( 2 , -1962866148 , 0 , 0 , 0 , 0 )" VT_BSTR
"MouseMove/-606( 1 , 0 , 145 , 35 )" VT_BSTR
```

Each line indicates an event, and the following information is provided: the name of the event, its identifier, and the list of parameters being passed to the event. The parameters that starts with = character, indicates a parameter by reference, in other words one that can changed during the event handler.

Now, we can see that the identifier for the BarParentChange event is 125, so we need to handle the Event event as:

```
// Notifies the application once the control fires an event.
void onEvent_Event(int _EventID)
{
   ;
   if ( _EventID == 125 ) /*event BarParentChange (Item as HITEM, Key as Variant, NewItem as HITEM, Cancel as Boolean) */
      cascadefile1.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 ( !cascadefile1.Items().EnableItem( cascadefile1.EventParam( 2 /*NewItem*/ ) ) )
            cascadefile1.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

event KeyDown (ByRef KeyCode as Integer, Shift as Integer)

Occurs when the user presses a key while an object has the focus.

Type	Description
KeyCode as Integer	(By Reference) An integer that represent the key code.
Shift as Integer	An integer that corresponds to the state of the SHIFT, CTRL, and ALT keys at the time of the event. The shift argument is a bit field with the least-significant bits corresponding to the SHIFT key (bit 0), the CTRL key (bit 1), and the ALT key (bit 2). These bits correspond to the values 1, 2, and 4, respectively. Some, all, or none of the bits can be set, indicating that some, all, or none of the keys are pressed. For example, if both CTRL and ALT are pressed, the value of shift is 6.

Use KeyDown and KeyUp event procedures if you need to respond to both the pressing and releasing of a key. You test for a condition by first assigning each result to a temporary integer variable and then comparing shift to a bit mask. Use the And operator with the shift argument to test whether the condition is greater than 0, indicating that the modifier was pressed, as in this example:

```
ShiftDown = (Shift And 1) > 0
CtrlDown = (Shift And 2) > 0
AltDown = (Shift And 4) > 0
```

In a procedure, you can test for any combination of conditions, as in this example:
If AltDown And CtrlDown Then

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

C#
```
private void KeyDown(object sender,ref short KeyCode,short Shift)
{
}
```

VB
```
Private Sub KeyDown(ByVal sender As System.Object,ByRef KeyCode As
Short,ByVal Shift As Short) Handles KeyDown
End Sub
```

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

C#

```
private void KeyDownEvent(object sender,
AxEXMILLERLib._ICascadeFileEvents_KeyDownEvent e)
{
}
```

C++

```
void OnKeyDown(short FAR* KeyCode,short Shift)
{
}
```

C++ Builder

```
void __fastcall KeyDown(TObject *Sender,short * KeyCode,short Shift)
{
}
```

Delphi

```
procedure KeyDown(ASender: TObject; var KeyCode : Smallint;Shift : Smallint);
begin
end;
```

Delphi 8 (.NET only)

```
procedure KeyDownEvent(sender: System.Object; e:
AxEXMILLERLib._ICascadeFileEvents_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
AxEXMILLERLib._ICascadeFileEvents_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(oCascadeFile,KeyCode,Shift)
RETURN
```

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

Java…
```
<SCRIPT EVENT="KeyDown(KeyCode,Shift)" LANGUAGE="JScript">
</SCRIPT>
```

VBSc…
```
<SCRIPT LANGUAGE="VBScript">
Function KeyDown(KeyCode,Shift)
End Function
</SCRIPT>
```

Visual Data…
```
Procedure OnComKeyDown Short llKeyCode Short llShift
    Forward Send OnComKeyDown llKeyCode llShift
End_Procedure
```

Visual Objects
```
METHOD OCX_KeyDown(KeyCode,Shift) CLASS MainDialog
RETURN NIL
```

X++
```
void onEvent_KeyDown(COMVariant /*short*/ _KeyCode,int _Shift)
{
}
```

XBasic
```
function KeyDown as v (KeyCode as N,Shift as N)
end function
```

dBASE
```
function nativeObject_KeyDown(KeyCode,Shift)
return
```

event KeyPress (ByRef KeyAscii as Integer)

Occurs when the user presses and releases an ANSI key.

Type	Description
KeyAscii as Integer	(By Reference) An integer that returns a standard numeric ANSI keycode.

The KeyPress event lets you immediately test keystrokes for validity or for formatting characters as they are typed. Changing the value of the keyascii argument changes the character displayed. Use KeyDown and KeyUp event procedures to handle any keystroke not recognized by KeyPress, such as function keys, editing keys, navigation keys, and any combinations of these with keyboard modifiers. Unlike the KeyDown and KeyUp events, KeyPress does not indicate the physical state of the keyboard; instead, it passes a character. KeyPress interprets the uppercase and lowercase of each character as separate key codes and, therefore, as two separate characters.

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

C#
```
private void KeyPress(object sender,ref short KeyAscii)
{
}
```

VB
```
Private Sub KeyPress(ByVal sender As System.Object,ByRef KeyAscii As Short)
Handles KeyPress
End Sub
```

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

C#
```
private void KeyPressEvent(object sender,
AxEXMILLERLib._ICascadeFileEvents_KeyPressEvent e)
{
}
```

C++
```
void OnKeyPress(short FAR* KeyAscii)
{
}
```

C++ Builder
```
void __fastcall KeyPress(TObject *Sender,short * KeyAscii)
{
}
```

Delphi
```
procedure KeyPress(ASender: TObject; var KeyAscii : Smallint);
begin
end;
```

Delphi 8 (.NET only)
```
procedure KeyPressEvent(sender: System.Object; e:
AxEXMILLERLib._ICascadeFileEvents_KeyPressEvent);
begin
end;
```

Powe...
```
begin event KeyPress(integer KeyAscii)
end event KeyPress
```

VB.NET
```
Private Sub KeyPressEvent(ByVal sender As System.Object, ByVal e As
AxEXMILLERLib._ICascadeFileEvents_KeyPressEvent) Handles KeyPressEvent
End Sub
```

VB6
```
Private Sub KeyPress(KeyAscii As Integer)
End Sub
```

VBA
```
Private Sub KeyPress(KeyAscii As Integer)
End Sub
```

VFP
```
LPARAMETERS KeyAscii
```

Xbas...
```
PROCEDURE OnKeyPress(oCascadeFile,KeyAscii)
RETURN
```

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

Java...
```
<SCRIPT EVENT="KeyPress(KeyAscii)" LANGUAGE="JScript">
</SCRIPT>
```

VBSc...
```
<SCRIPT LANGUAGE="VBScript">
Function KeyPress(KeyAscii)
End Function
</SCRIPT>
```

Visual Data...

```
Procedure OnComKeyPress Short llKeyAscii
    Forward Send OnComKeyPress llKeyAscii
End_Procedure
```

Visual Objects

```
METHOD OCX_KeyPress(KeyAscii) CLASS MainDialog
RETURN NIL
```

X++

```
void onEvent_KeyPress(COMVariant /*short*/ _KeyAscii)
{
}
```

XBasic

```
function KeyPress as v (KeyAscii as N)
end function
```

dBASE

```
function nativeObject_KeyPress(KeyAscii)
return
```

event KeyUp (ByRef KeyCode as Integer, Shift as Integer)

Occur when the user releases a key while an object has the focus.

Type	Description
KeyCode as Integer	(By Reference) An integer that represent the key code.
Shift as Integer	An integer that corresponds to the state of the SHIFT, CTRL, and ALT keys at the time of the event. The shift argument is a bit field with the least-significant bits corresponding to the SHIFT key (bit 0), the CTRL key (bit 1), and the ALT key (bit 2). These bits correspond to the values 1, 2, and 4, respectively. Some, all, or none of the bits can be set, indicating that some, all, or none of the keys are pressed. For example, if both CTRL and ALT are pressed, the value of shift is 6.

Use the KeyUp event procedure to respond to the releasing of a key.

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

C#
```
private void KeyUp(object sender,ref short KeyCode,short Shift)
{
}
```

VB
```
Private Sub KeyUp(ByVal sender As System.Object,ByRef KeyCode As Short,ByVal
Shift As Short) Handles KeyUp
End Sub
```

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

C#
```
private void KeyUpEvent(object sender,
AxEXMILLERLib._ICascadeFileEvents_KeyUpEvent e)
{
}
```

C++
```
void OnKeyUp(short FAR* KeyCode,short Shift)
{
}
```

C++ Builder
```
void __fastcall KeyUp(TObject *Sender,short * KeyCode,short Shift)
```

```
{
}
```

Delphi
```
procedure KeyUp(ASender: TObject; var KeyCode : Smallint;Shift : Smallint);
begin
end;
```

Delphi 8 (.NET only)
```
procedure KeyUpEvent(sender: System.Object; e:
AxEXMILLERLib._ICascadeFileEvents_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
AxEXMILLERLib._ICascadeFileEvents_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(oCascadeFile,KeyCode,Shift)
RETURN
```

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

Java...
```
<SCRIPT EVENT="KeyUp(KeyCode,Shift)" LANGUAGE="JScript">
</SCRIPT>
```

VBSc...
```
<SCRIPT LANGUAGE="VBScript">
Function KeyUp(KeyCode,Shift)
```

```
End Function
</SCRIPT>
```

Visual Data...

```
Procedure OnComKeyUp Short llKeyCode Short llShift
    Forward Send OnComKeyUp llKeyCode llShift
End_Procedure
```

Visual Objects

```
METHOD OCX_KeyUp(KeyCode,Shift) CLASS MainDialog
RETURN NIL
```

X++

```
void onEvent_KeyUp(COMVariant /*short*/ _KeyCode,int _Shift)
{
}
```

XBasic

```
function KeyUp as v (KeyCode as N,Shift as N)
end function
```

dBASE

```
function nativeObject_KeyUp(KeyCode,Shift)
return
```

event MouseDown (Button as Integer, Shift as Integer, X as OLE_XPOS_PIXELS, Y as OLE_YPOS_PIXELS)

Occur when the user presses a mouse button.

Type	Description
Button as Integer	An integer that identifies the button that was pressed to cause the event.
Shift as Integer	An integer that corresponds to the state of the SHIFT, CTRL, and ALT keys when the button specified in the button argument is pressed or released.
X as OLE_XPOS_PIXELS	A single that specifies the current X location of the mouse pointer. The X value is always expressed in container coordinates.
Y as OLE_YPOS_PIXELS	A single that specifies the current Y location of the mouse pointer. The Y value is always expressed in container coordinates.

Use the MouseDown or MouseUp event procedure to specify actions that will occur when a mouse button is pressed or released. Unlike the Click and DblClick events, MouseDown and MouseUp events lets you distinguish between the left, right, and middle mouse buttons. You can also write code for mouse-keyboard combinations that use the SHIFT, CTRL, and ALT keyboard modifiers.

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

C#

```
private void MouseDownEvent(object sender,short Button,short Shift,int X,int Y)
{
}
```

VB

```
Private Sub MouseDownEvent(ByVal sender As System.Object,ByVal Button As Short,ByVal Shift As Short,ByVal X As Integer,ByVal Y As Integer) Handles MouseDownEvent
End Sub
```

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

C#

```
private void MouseDownEvent(object sender, AxEXMILLERLib._ICascadeFileEvents_MouseDownEvent e)
{
}
```

C++

```
void OnMouseDown(short Button,short Shift,long X,long Y)
{
}
```

C++ Builder

```
void __fastcall MouseDown(TObject *Sender,short Button,short Shift,int X,int Y)
{
}
```

Delphi

```
procedure MouseDown(ASender: TObject; Button : Smallint;Shift : Smallint;X :
Integer;Y : Integer);
begin
end;
```

Delphi 8 (.NET only)

```
procedure MouseDownEvent(sender: System.Object; e:
AxEXMILLERLib._ICascadeFileEvents_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
AxEXMILLERLib._ICascadeFileEvents_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(oCascadeFile,Button,Shift,X,Y)
RETURN
```

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

Java…

```
<SCRIPT EVENT="MouseDown(Button,Shift,X,Y)" LANGUAGE="JScript">
</SCRIPT>
```

VBSc…

```
<SCRIPT LANGUAGE="VBScript">
Function MouseDown(Button,Shift,X,Y)
End Function
</SCRIPT>
```

Visual Data…

```
Procedure OnComMouseDown Short llButton Short llShift OLE_XPOS_PIXELS llX
OLE_YPOS_PIXELS llY
   Forward Send OnComMouseDown llButton llShift llX llY
End_Procedure
```

Visual Objects

```
METHOD OCX_MouseDown(Button,Shift,X,Y) CLASS MainDialog
RETURN NIL
```

X++

```
void onEvent_MouseDown(int _Button,int _Shift,int _X,int _Y)
{
}
```

XBasic

```
function MouseDown as v (Button as N,Shift as N,X as
OLE::Exontrol.CascadeFile.1::OLE_XPOS_PIXELS,Y as
OLE::Exontrol.CascadeFile.1::OLE_YPOS_PIXELS)
end function
```

dBASE

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

event MouseMove (Button as Integer, Shift as Integer, X as OLE_XPOS_PIXELS, Y as OLE_YPOS_PIXELS)

Occurs when the user moves the mouse.

Type	Description
Button as Integer	An integer that corresponds to the state of the mouse buttons in which a bit is set if the button is down.
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.

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

C#
```
private void MouseMoveEvent(object sender,short Button,short Shift,int X,int Y)
{
}
```

VB
```
Private Sub MouseMoveEvent(ByVal sender As System.Object,ByVal Button As
Short,ByVal Shift As Short,ByVal X As Integer,ByVal Y As Integer) Handles
MouseMoveEvent
End Sub
```

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

C#
```
private void MouseMoveEvent(object sender,
AxEXMILLERLib._ICascadeFileEvents_MouseMoveEvent e)
{
}
```

C++
```
void OnMouseMove(short Button,short Shift,long X,long Y)
```

```
{
}
```

C++ Builder

```
void __fastcall MouseMove(TObject *Sender,short Button,short Shift,int X,int Y)
{
}
```

Delphi

```
procedure MouseMove(ASender: TObject; Button : Smallint;Shift : Smallint;X :
Integer;Y : Integer);
begin
end;
```

Delphi 8 (.NET only)

```
procedure MouseMoveEvent(sender: System.Object; e:
AxEXMILLERLib._ICascadeFileEvents_MouseMoveEvent);
begin
end;
```

Powe…

```
begin event MouseMove(integer Button,integer Shift,long X,long Y)
end event MouseMove
```

VB.NET

```
Private Sub MouseMoveEvent(ByVal sender As System.Object, ByVal e As
AxEXMILLERLib._ICascadeFileEvents_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(oCascadeFile,Button,Shift,X,Y)
RETURN
```

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

Java…

```
<SCRIPT EVENT="MouseMove(Button,Shift,X,Y)" LANGUAGE="JScript">
</SCRIPT>
```

VBSc…

```
<SCRIPT LANGUAGE="VBScript">
Function MouseMove(Button,Shift,X,Y)
End Function
</SCRIPT>
```

Visual Data…

```
Procedure OnComMouseMove Short llButton Short llShift OLE_XPOS_PIXELS llX
OLE_YPOS_PIXELS llY
    Forward Send OnComMouseMove llButton llShift llX llY
End_Procedure
```

Visual Objects

```
METHOD OCX_MouseMove(Button,Shift,X,Y) CLASS MainDialog
RETURN NIL
```

X++

```
void onEvent_MouseMove(int _Button,int _Shift,int _X,int _Y)
{
}
```

XBasic

```
function MouseMove as v (Button as N,Shift as N,X as
OLE::Exontrol.CascadeFile.1::OLE_XPOS_PIXELS,Y as
OLE::Exontrol.CascadeFile.1::OLE_YPOS_PIXELS)
end function
```

dBASE

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

event MouseUp (Button as Integer, Shift as Integer, X as OLE_XPOS_PIXELS, Y as OLE_YPOS_PIXELS)

Occurs when the user releases a mouse button.

Type	Description
Button as Integer	An integer that identifies the button that was pressed to cause the event.
Shift as Integer	An integer that corresponds to the state of the SHIFT, CTRL, and ALT keys when the button specified in the button argument is pressed or released.
X as OLE_XPOS_PIXELS	A single that specifies the current X location of the mouse pointer. The x values is always expressed in container coordinates.
Y as OLE_YPOS_PIXELS	A single that specifies the current Y location of the mouse pointer. The y values is always expressed in container coordinates.

Use a MouseDown or MouseUp event procedure to specify actions that will occur when a mouse button is pressed or released. Unlike the Click and DblClick events, MouseDown and MouseUp events lets you distinguish between the left, right, and middle mouse buttons. You can also write code for mouse-keyboard combinations that use the SHIFT, CTRL, and ALT keyboard modifiers.

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

C#
```
private void MouseUpEvent(object sender,short Button,short Shift,int X,int Y)
{
}
```

VB
```
Private Sub MouseUpEvent(ByVal sender As System.Object,ByVal Button As Short,ByVal Shift As Short,ByVal X As Integer,ByVal Y As Integer) Handles MouseUpEvent
End Sub
```

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

C#
```
private void MouseUpEvent(object sender, AxEXMILLERLib._ICascadeFileEvents_MouseUpEvent e)
{
}
```

C++

```
void OnMouseUp(short Button,short Shift,long X,long Y)
{
}
```

C++ Builder

```
void __fastcall MouseUp(TObject *Sender,short Button,short Shift,int X,int Y)
{
}
```

Delphi

```
procedure MouseUp(ASender: TObject; Button : Smallint;Shift : Smallint;X :
Integer;Y : Integer);
begin
end;
```

Delphi 8 (.NET only)

```
procedure MouseUpEvent(sender: System.Object; e:
AxEXMILLERLib._ICascadeFileEvents_MouseUpEvent);
begin
end;
```

Powe…

```
begin event MouseUp(integer Button,integer Shift,long X,long Y)
end event MouseUp
```

VB.NET

```
Private Sub MouseUpEvent(ByVal sender As System.Object, ByVal e As
AxEXMILLERLib._ICascadeFileEvents_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(oCascadeFile,Button,Shift,X,Y)
```

```
RETURN
```

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

Java…

```
<SCRIPT EVENT="MouseUp(Button,Shift,X,Y)" LANGUAGE="JScript">
</SCRIPT>
```

VBSc…

```
<SCRIPT LANGUAGE="VBScript">
Function MouseUp(Button,Shift,X,Y)
End Function
</SCRIPT>
```

Visual Data…

```
Procedure OnComMouseUp Short llButton Short llShift OLE_XPOS_PIXELS llX
OLE_YPOS_PIXELS llY
    Forward Send OnComMouseUp llButton llShift llX llY
End_Procedure
```

Visual Objects

```
METHOD OCX_MouseUp(Button,Shift,X,Y) CLASS MainDialog
RETURN NIL
```

X++

```
void onEvent_MouseUp(int _Button,int _Shift,int _X,int _Y)
{
}
```

XBasic

```
function MouseUp as v (Button as N,Shift as N,X as
OLE::Exontrol.CascadeFile.1::OLE_XPOS_PIXELS,Y as
OLE::Exontrol.CascadeFile.1::OLE_YPOS_PIXELS)
end function
```

dBASE

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

event RClick ()

Occurs once the user right clicks the control.

Type	Description

Notifies your application once the user right-clicks the control. Use the MouseDown or MouseUp event procedure to specify actions that will occur when a mouse button is pressed or released.

event ViewEndChanging (Operation as ViewOperationEnum)

Occurs once the user is about to change the view.

Type	Description
Operation as ViewOperationEnum	A ViewOperationEnum expression that specifies the operation that ended.

The ViewStartChanging / ViewEndChanging events notify your application that an operation starts or ends. For instance, ViewStartChanging(exSelectItemChange) / ViewEndChanging(exSelectItemChange) events notify your application that a file has been selected.

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

C#
```
private void ViewEndChanging(object
sender,exontrol.EXMILLERLib.ViewOperationEnum   Operation)
{
}
```

VB
```
Private Sub ViewEndChanging(ByVal sender As System.Object,ByVal Operation As
exontrol.EXMILLERLib.ViewOperationEnum) Handles ViewEndChanging
End Sub
```

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

C#
```
private void ViewEndChanging(object sender,
AxEXMILLERLib._ICascadeFileEvents_ViewEndChangingEvent e)
{
}
```

C++
```
void OnViewEndChanging(long   Operation)
{
}
```

C++ Builder
```
void __fastcall ViewEndChanging(TObject
*Sender,Exmillerlib_tlb::ViewOperationEnum   Operation)
{
}
```

Delphi

```
procedure ViewEndChanging(ASender: TObject; Operation : ViewOperationEnum);
begin
end;
```

Delphi 8 (.NET only)

```
procedure ViewEndChanging(sender: System.Object; e:
AxEXMILLERLib._ICascadeFileEvents_ViewEndChangingEvent);
begin
end;
```

Powe…

```
begin event ViewEndChanging(long  Operation)


end event ViewEndChanging
```

VB.NET

```
Private Sub ViewEndChanging(ByVal sender As System.Object, ByVal e As
AxEXMILLERLib._ICascadeFileEvents_ViewEndChangingEvent) Handles
ViewEndChanging
End Sub
```

VB6

```
Private Sub ViewEndChanging(ByVal Operation As
EXMILLERLibCtl.ViewOperationEnum)
End Sub
```

VBA

```
Private Sub ViewEndChanging(ByVal Operation As Long)
End Sub
```

VFP

```
LPARAMETERS Operation
```

Xbas…

```
PROCEDURE OnViewEndChanging(oCascadeFile,Operation)


RETURN
```

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

Java…

```
<SCRIPT EVENT="ViewEndChanging(Operation)" LANGUAGE="JScript">
</SCRIPT>
```

VBSc…

```
<SCRIPT LANGUAGE="VBScript">
Function ViewEndChanging(Operation)
End Function
</SCRIPT>
```

Visual Data…

```
Procedure OnComViewEndChanging OLEViewOperationEnum   llOperation
    Forward Send OnComViewEndChanging llOperation
End_Procedure
```

Visual Objects

```
METHOD OCX_ViewEndChanging(Operation) CLASS MainDialog
RETURN NIL
```

X++

```
void onEvent_ViewEndChanging(int   _Operation)
{
}
```

XBasic

```
function ViewEndChanging as v (Operation  as
OLE::Exontrol.ExMiller.1::ViewOperationEnum)
end function
```

dBASE

```
function nativeObject_ViewEndChanging(Operation)
return
```

event ViewStartChanging (Operation as ViewOperationEnum)

Occurs once the user is about to change the view.

Type	Description
Operation as ViewOperationEnum	A ViewOperationEnum expression that specifies the operation is about to begin.

The ViewStartChanging / ViewEndChanging events notify your application that an operation starts or ends. For instance, ViewStartChanging(exSelectItemChange) / ViewEndChanging(exSelectItemChange) events notify your application that a file has been selected.

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

C#
```
private void ViewStartChanging(object
sender,exontrol.EXMILLERLib.ViewOperationEnum   Operation)
{
}
```

VB
```
Private Sub ViewStartChanging(ByVal sender As System.Object,ByVal Operation As
exontrol.EXMILLERLib.ViewOperationEnum) Handles ViewStartChanging
End Sub
```

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

C#
```
private void ViewStartChanging(object sender,
AxEXMILLERLib._ICascadeFileEvents_ViewStartChangingEvent e)
{
}
```

C++
```
void OnViewStartChanging(long   Operation)
{
}
```

C++ Builder
```
void __fastcall ViewStartChanging(TObject
*Sender,Exmillerlib_tlb::ViewOperationEnum   Operation)
{
}
```

Delphi

```
procedure ViewStartChanging(ASender: TObject; Operation : ViewOperationEnum);
begin
end;
```

Delphi 8 (.NET only)

```
procedure ViewStartChanging(sender: System.Object; e:
AxEXMILLERLib._ICascadeFileEvents_ViewStartChangingEvent);
begin
end;
```

Powe…

```
begin event ViewStartChanging(long  Operation)

end event ViewStartChanging
```

VB.NET

```
Private Sub ViewStartChanging(ByVal sender As System.Object, ByVal e As
AxEXMILLERLib._ICascadeFileEvents_ViewStartChangingEvent) Handles
ViewStartChanging
End Sub
```

VB6

```
Private Sub ViewStartChanging(ByVal Operation As
EXMILLERLibCtl.ViewOperationEnum)
End Sub
```

VBA

```
Private Sub ViewStartChanging(ByVal Operation As Long)
End Sub
```

VFP

```
LPARAMETERS Operation
```

Xbas…

```
PROCEDURE OnViewStartChanging(oCascadeFile,Operation)

RETURN
```

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

Java…

```
<SCRIPT EVENT="ViewStartChanging(Operation)" LANGUAGE="JScript">
</SCRIPT>
```

VBSc…

```
<SCRIPT LANGUAGE="VBScript">
Function ViewStartChanging(Operation)
End Function
</SCRIPT>
```

Visual Data…

```
Procedure OnComViewStartChanging OLEViewOperationEnum   llOperation
    Forward Send OnComViewStartChanging llOperation
End_Procedure
```

Visual Objects

```
METHOD OCX_ViewStartChanging(Operation) CLASS MainDialog
RETURN NIL
```

X++

```
void onEvent_ViewStartChanging(int   _Operation)
{
}
```

XBasic

```
function ViewStartChanging as v (Operation  as
OLE::Exontrol.ExMiller.1::ViewOperationEnum)
end function
```

dBASE

```
function nativeObject_ViewStartChanging(Operation)
return
```

Expressions

An expression is a string which defines a formula or criteria, that's evaluated at runtime. The expression may be a combination of variables, constants, strings, dates and operators/functions. For instance 1000 format `` gets 1,000.00 for US format, while 1.000,00 is displayed for German format.

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

Usage examples:

- 100 + 200, adds two numbers and returns 300
- "100" + 200, concatenates the strings, and returns "100200"
- currency(1000) displays the value in currency format based on the current regional setting, such as "$1,000.00" for US format.
- 1000 format `` gets 1,000.00 for English format, while 1.000,00 is displayed for German format
- 1000 format `2|.|3|,` always gets 1,000.00 no matter of settings in the control panel.
- date(value) format `MMM d, yyyy` , returns the date such as Sep 2, 2023, for English format
- upper("string") converts the giving string in uppercase letters, such as "STRING"
- date(dateS('3/1/' + year(9:=#1/1/2018#)) + ((1:=(((255 - 11 * (year(=:9) mod 19)) - 21) mod 30) + 21) + (=:1 > 48 ? -1 : 0) + 6 - ((year(=:9) + int(year(=:9) / 4)) + =:1 + (=:1 > 48 ? -1 : 0) + 1) mod 7)) returns the date the Easter Sunday will fall, for year 2018. In this case the expression returns #4/1/2018#. If #1/1/2018# is replaced with #1/1/2019#, the expression returns #4/21/2019#.

Listed bellow are all predefined constants, operators and functions the general-expression supports:

The constants can be represented as:

- numbers in **decimal** format (where dot character specifies the decimal separator). For instance: -1, 100, 20.45, .99 and so on
- numbers in **hexa-decimal** format (preceded by **0x** or **0X** sequence), uses sixteen distinct symbols, most often the symbols 0-9 to represent values zero to nine, and A, B, C, D, E, F (or alternatively a, b, c, d, e, f) to represent values ten to fifteen. Hexadecimal numerals are widely used by computer system designers and

programmers. As each hexadecimal digit represents four binary digits (bits), it allows a more human-friendly representation of binary-coded values. For instance, 0xFF, 0x00FF00, and so so.
- **date-time** in format **#mm/dd/yyyy hh:mm:ss#**, For instance, #1/31/2001 10:00# means the January 31th, 2001, 10:00 AM
- **string**, if it starts / ends with any of the **'** or **`** or **"** characters. If you require the starting character inside the string, it should be escaped ( preceded by a \ character ). For instance, `Mihai`, "Filimon", 'has', "\"a quote\"", and so on

The predefined constants are:

- **bias** (BIAS constant), defines the difference, in minutes, between Coordinated Universal Time (UTC) and local time. For example, Middle European Time (MET, GMT+01:00) has a time zone bias of "-60" because it is one hour ahead of UTC. Pacific Standard Time (PST, GMT-08:00) has a time zone bias of "+480" because it is eight hours behind UTC. For instance, date(value - bias/24/60) converts the UTC time to local time, or date(date('now') + bias/24/60) converts the current local time to UTC time. For instance, "date(value - bias/24/60)" converts the value date-time from UTC to local time, while "date(value + bias/24/60)" converts the local-time to UTC time.
- **dpi** (DPI constant), specifies the current DPI setting. and it indicates the minimum value between **dpix** and **dpiy** constants. For instance, if current DPI setting is 100%, the dpi constant returns 1, if 150% it returns 1.5, and so on. For instance, the expression value * dpi returns the value if the DPI setting is 100%, or value * 1.5 in case, the DPI setting is 150%
- **dpix** (DPIX constant), specifies the current DPI setting on x-scale. For instance, if current DPI setting is 100%, the dpix constant returns 1, if 150% it returns 1.5, and so on. For instance, the expression value * dpix returns the value if the DPI setting is 100%, or value * 1.5 in case, the DPI setting is 150%
- **dpiy** (DPIY constant), specifies the current DPI setting on x-scale. For instance, if current DPI setting is 100%, the dpiy constant returns 1, if 150% it returns 1.5, and so on. For instance, the expression value * dpiy returns the value if the DPI setting is 100%, or value * 1.5 in case, the DPI setting is 150%

The supported binary arithmetic operators are:

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

The supported unary boolean operators are:

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

The supported binary boolean operators are:

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

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

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

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

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

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

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

  variable := expression

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

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

  =: variable

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

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

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

expression ? true_part : false_part

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

The *in*, *switch* and *case()* use binary search to look for elements so they are faster then using iif and or expressions. Obviously, the priority of the operations inside the expression is determined by ( ) parenthesis and the priority for each operator.

The supported conversion unary operators are:

- **type** (unary operator) retrieves the type of the object. The type operator may return any of the following: 0 - empty (not initialized), 1 - null, 2 - short, 3 - long, 4 - float, 5 - double, 6 - currency, **7 - date**, **8 - string**, 9 - object, 10 - error, **11 - boolean**, 12 - variant, 13 - any, 14 - decimal, 16 - char, 17 - byte, 18 - unsigned short, 19 - unsigned long, 20 - long on 64 bits, 21 - unsigned long on 64 bites. For instance *type(%1) = 8* specifies the cells (on the column with the index 1) that contains string values.
- **str** (unary operator) converts the expression to a string. The str operator converts the expression to a string. For instance, the *str(-12.54)* returns the string "-12.54".
- **dbl** (unary operator) converts the expression to a number. The dbl operator converts the expression to a number. For instance, the *dbl("12.54")* returns 12.54
- **date** (unary operator) converts the expression to a date, based on your regional settings. For instance, the *date(``)* gets the current date (no time included), the *date(`now`)* gets the current date-time, while the *date("01/01/2001")* returns #1/1/2001#
- **dateS** (unary operator) converts the string expression to a date using the format MM/DD/YYYY HH:MM:SS. For instance, the *dateS("01/01/2001 14:00:00")* returns #1/1/2001 14:00:00#
- **hex** (unary operator) converts the giving string from hexa-representation to a numeric value, or converts the giving numeric value to hexa-representation as string. For instance, hex(`FF`) returns 255, while the hex(255) or hex(0xFF) returns the `FF` string. The hex(hex(`FFFFFFFF`)) always returns `FFFFFFFF` string, as the second hex call converts the giving string to a number, and the first hex call converts the returned number to string representation (hexa-representation).

The bitwise operators for numbers are:

- a **bitand** b (binary operator) computes the AND operation on bits of a and b, and returns the unsigned value. For instance, 0x01001000 bitand 0x10111000 returns 0x00001000.
- a **bitor** b (binary operator) computes the OR operation on bits of a and b, and returns the unsigned value. For instance, 0x01001000 bitor 0x10111000 returns 0x11111000.
- a **bitxor** b (binary operator) computes the XOR (exclusive-OR) operation on bits of a and b, and returns the unsigned value. For instance, 0x01110010 bitxor 0x10101010 returns 0x11011000.

- a **bitshift** (b) (binary operator) shifts every bit of a value to the left if b is negative, or to the right if b is positive, for b times, and returns the unsigned value. For instance, 128 bitshift 1 returns 64 (dividing by 2) or 128 bitshift (-1) returns 256 (multiplying by 2)
- **bitnot** (unary operator) flips every bit of x, and returns the unsigned value. For instance, bitnot(0x00FF0000) returns 0xFF00FFFF.

The operators for numbers are:

- **int** (unary operator) retrieves the integer part of the number. For instance, the *int(12.54)* returns 12
- **round** (unary operator) rounds the number ie 1.2 gets 1, since 1.8 gets 2. For instance, the *round(12.54)* returns 13
- **floor** (unary operator) returns the largest number with no fraction part that is not greater than the value of its argument. For instance, the *floor(12.54)* returns 12
- **abs** (unary operator) retrieves the absolute part of the number ie -1 gets 1, 2 gets 2. For instance, the *abs(-12.54)* returns 12.54
- **sin** (unary operator) returns the sine of an angle of x radians. For instance, the *sin(3.14)* returns 0.001593.
- **cos** (unary operator) returns the cosine of an angle of x radians. For instance, the *cos(3.14)* returns -0.999999.
- **asin** (unary operator) returns the principal value of the arc sine of x, expressed in radians. For instance, the *2*asin(1)* returns the value of PI.
- **acos** (unary operator) returns the principal value of the arc cosine of x, expressed in radians. For instance, the *2*acos(0)* returns the value of PI
- **sqrt** (unary operator) returns the square root of x. For instance, the *sqrt(81)* returns 9.
- **currency** (unary operator) formats the giving number as a currency string, as indicated by the control panel. For instance, *currency(value)* displays the value using the current format for the currency ie, 1000 gets displayed as $1,000.00, for US format.
- value **format** 'flags' (binary operator) formats a numeric value with specified flags. The format method formats numeric or date expressions (depends on the type of the value, explained at operators for dates). 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 the settings in your control panel. If formatting the number fails for some invalid parameter, the value is displayed with no formatting.

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

  - *NumDigits* - specifies the number of fractional digits, If the flag is missing, the

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

The operators for strings are:

- **len** (unary operator) retrieves the number of characters in the string. For instance, the *len("Mihai")* returns 5.
- **lower** (unary operator) returns a string expression in lowercase letters. For instance, the *lower("MIHAI")* returns "mihai"
- **upper** (unary operator) returns a string expression in uppercase letters. For instance, the *upper("mihai")* returns "MIHAI"
- **proper** (unary operator) returns from a character expression a string capitalized as appropriate for proper names. For instance, the *proper("mihai")* returns "Mihai"
- **ltrim** (unary operator) removes spaces on the left side of a string. For instance, the *ltrim(" mihai")* returns "mihai"
- **rtrim** (unary operator) removes spaces on the right side of a string. For instance, the *rtrim("mihai ")* returns "mihai"

- **trim** (unary operator) removes spaces on both sides of a string. For instance, the *trim(" mihai ")* returns "mihai"
- **reverse** (unary operator) reverses the order of the characters in the string a. For instance, the *reverse("Mihai")* returns "iahiM"
- a **startwith** b (binary operator) specifies whether a string starts with specified string (0 if not found, -1 if found). For instance *"Mihai" startwith "Mi"* returns -1
- a **endwith** b (binary operator) specifies whether a string ends with specified string (0 if not found, -1 if found). For instance *"Mihai" endwith "ai"* returns -1
- a **contains** b (binary operator) specifies whether a string contains another specified string (0 if not found, -1 if found). For instance *"Mihai" contains "ha"* returns -1
- a **left** b (binary operator) retrieves the left part of the string. For instance *"Mihai" left 2* returns "Mi".
- a **right** b (binary operator) retrieves the right part of the string. For instance *"Mihai" right 2* returns "ai"
- a **lfind** b (binary operator) The a lfind b (binary operator) searches the first occurrence of the string b within string a, and returns -1 if not found, or the position of the result (zero-index). For instance *"ABCABC" lfind "C"* returns 2
- a **rfind** b (binary operator) The a rfind b (binary operator) searches the last occurrence of the string b within string a, and returns -1 if not found, or the position of the result (zero-index). For instance *"ABCABC" rfind "C"* returns 5.
- a **mid** b (binary operator) retrieves the middle part of the string a starting from b (1 means first position, and so on). For instance *"Mihai" mid 2* returns "ihai"
- a **count** b (binary operator) retrieves the number of occurrences of the b in a. For instance *"Mihai" count "i"* returns 2.
- a **replace** b **with** c (double binary operator) replaces in a the b with c, and gets the result. For instance, the *"Mihai" replace "i" with ""* returns "Mha" string, as it replaces all "i" with nothing.
- a **split** b (binary operator) splits the a using the separator b, and returns an array. For instance, the *weekday(value) array 'Sun Mon Thu Wed Thu Fri Sat' split ' '* gets the weekday as string. This operator can be used with the array.
- a **like** b (binary operator) compares the string a against the pattern b. The pattern b may contain wild-characters such as *, ?, # or [] and can have multiple patterns separated by space character. In order to have the space, or any other wild-character inside the pattern, it has to be escaped, or in other words it should be preceded by a \ character. For instance value like `F*e` matches all strings that start with F and ends on e, or value like `a* b*` indicates any strings that start with a or b character.
- a **lpad** b (binary operator) pads the value of a to the left with b padding pattern. For instance, 12 lpad "0000" generates the string "0012".
- a **rpad** b (binary operator) pads the value of a to the right with b padding pattern. For instance, 12 lpad "____" generates the string "12__".
- a **concat** b (binary operator) concatenates the a (as string) for b times. For instance, "x" concat 5, generates the string "xxxxx".

The operators for dates are:

- **time** (unary operator) retrieves the time of the date in string format, as specified in the control's panel. For instance, the *time(#1/1/2001 13:00#)* returns "1:00:00 PM"
- **timeF** (unary operator) retrieves the time of the date in string format, as "HH:MM:SS". For instance, the *timeF(#1/1/2001 13:00#)* returns "13:00:00"
- **shortdate** (unary operator) formats a date as a date string using the short date format, as specified in the control's panel. For instance, the *shortdate(#1/1/2001 13:00#)* returns "1/1/2001"
- **shortdateF** (unary operator) formats a date as a date string using the "MM/DD/YYYY" format. For instance, the *shortdateF(#1/1/2001 13:00#)* returns "01/01/2001"
- **dateF** (unary operator) converts the date expression to a string expression in "MM/DD/YYYY HH:MM:SS" format. For instance, the *dateF(#01/01/2001 14:00:00#)* returns #01/01/2001 14:00:00#
- **longdate** (unary operator) formats a date as a date string using the long date format, as specified in the control's panel. For instance, the *longdate(#1/1/2001 13:00#)* returns "Monday, January 01, 2001"
- **year** (unary operator) retrieves the year of the date (100,...,9999). For instance, the *year(#12/31/1971 13:14:15#)* returns 1971
- **month** (unary operator) retrieves the month of the date (1, 2,...,12). For instance, the *month(#12/31/1971 13:14:15#)* returns 12.
- **day** (unary operator) retrieves the day of the date (1, 2,...,31). For instance, the *day(#12/31/1971 13:14:15#)* returns 31
- **yearday** (unary operator) retrieves the number of the day in the year, or the days since January 1st (0, 1,...,365). For instance, the *yearday(#12/31/1971 13:14:15#)* returns 365
- **weekday** (unary operator) retrieves the number of days since Sunday (0 - Sunday, 1 - Monday,..., 6 - Saturday). For instance, the *weekday(#12/31/1971 13:14:15#)* returns 5.
- **hour** (unary operator) retrieves the hour of the date (0, 1, ..., 23). For instance, the *hour(#12/31/1971 13:14:15#)* returns 13
- **min** (unary operator) retrieves the minute of the date (0, 1, ..., 59). For instance, the *min(#12/31/1971 13:14:15#)* returns 14
- **sec** (unary operator) retrieves the second of the date (0, 1, ..., 59). For instance, the *sec(#12/31/1971 13:14:15#)* returns 15
- value **format** 'flags' (binary operator) formats a date expression with specified flags. The format method formats numeric (depends on the type of the value, explained at operators for numbers) or date expressions. If not supported, the value is formatted as a number (the date format is supported by newer version only). The flags specifies the format picture string that is used to form the date. Possible values for the format picture string are defined below. For instance, the *date(value) format `MMM d, yyyy`*

returns "Sep 2, 2023"

The following table defines the format types used to represent days:

- d, day of the month as digits without leading zeros for single-digit days (8)
- dd, day of the month as digits with leading zeros for single-digit days (08)
- ddd, abbreviated day of the week as specified by the current locale ("Mon" in English)
- dddd, day of the week as specified by the current locale ("Monday" in English)

The following table defines the format types used to represent months:

- M, month as digits without leading zeros for single-digit months (4)
- MM, month as digits with leading zeros for single-digit months (04)
- MMM, abbreviated month as specified by the current locale ("Nov" in English)
- MMMM, month as specified by the current locale ("November" for English)

The following table defines the format types used to represent years:

- y, year represented only by the last digit (3)
- yy, year represented only by the last two digits. A leading zero is added for single-digit years (03)
- yyy, year represented 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.
- yyyy, behaves identically to "yyyy"

The following table defines the format types used to represent era:

- g, period/era string formatted as specified by the CAL_SERASTRING value (ignored if there is no associated era or period string)
- gg, period/era string formatted as specified by the CAL_SERASTRING value (ignored if there is no associated era or period string)

The following table defines the format types used to represent hours:

- h, hours with no leading zero for single-digit hours; 12-hour clock
- hh, hours with leading zero for single-digit hours; 12-hour clock
- H, hours with no leading zero for single-digit hours; 24-hour clock

- HH, hours with leading zero for single-digit hours; 24-hour clock

The following table defines the format types used to represent minutes:

- m, minutes with no leading zero for single-digit minutes
- mm, minutes with leading zero for single-digit minutes

The following table defines the format types used to represent seconds:

- s, seconds with no leading zero for single-digit seconds
- ss, seconds with leading zero for single-digit seconds

The following table defines the format types used to represent time markers:

- t, one character time marker string, such as A or P
- tt, multi-character time marker string, such as AM or PM

The expression supports also **immediate if** (similar with iif in visual basic, or ? : in C++) ie cond ? value_true : value_false, which means that once that cond is true the value_true is used, else the value_false is used. Also, it supports variables, up to 10 from 0 to 9. For instance, 0:="Abc" means that in the variable 0 is "Abc", and =:0 means retrieves the value of the variable 0. For instance, the len(%0) ? (0:=(%1+%2) ? currency(=:0) else `` ) : `` gets the sum between second and third column in currency format if it is not zero, and only if the first column is not empty. As you can see you can use the variables to avoid computing several times the same thing (in this case the sum %1 and %2 .