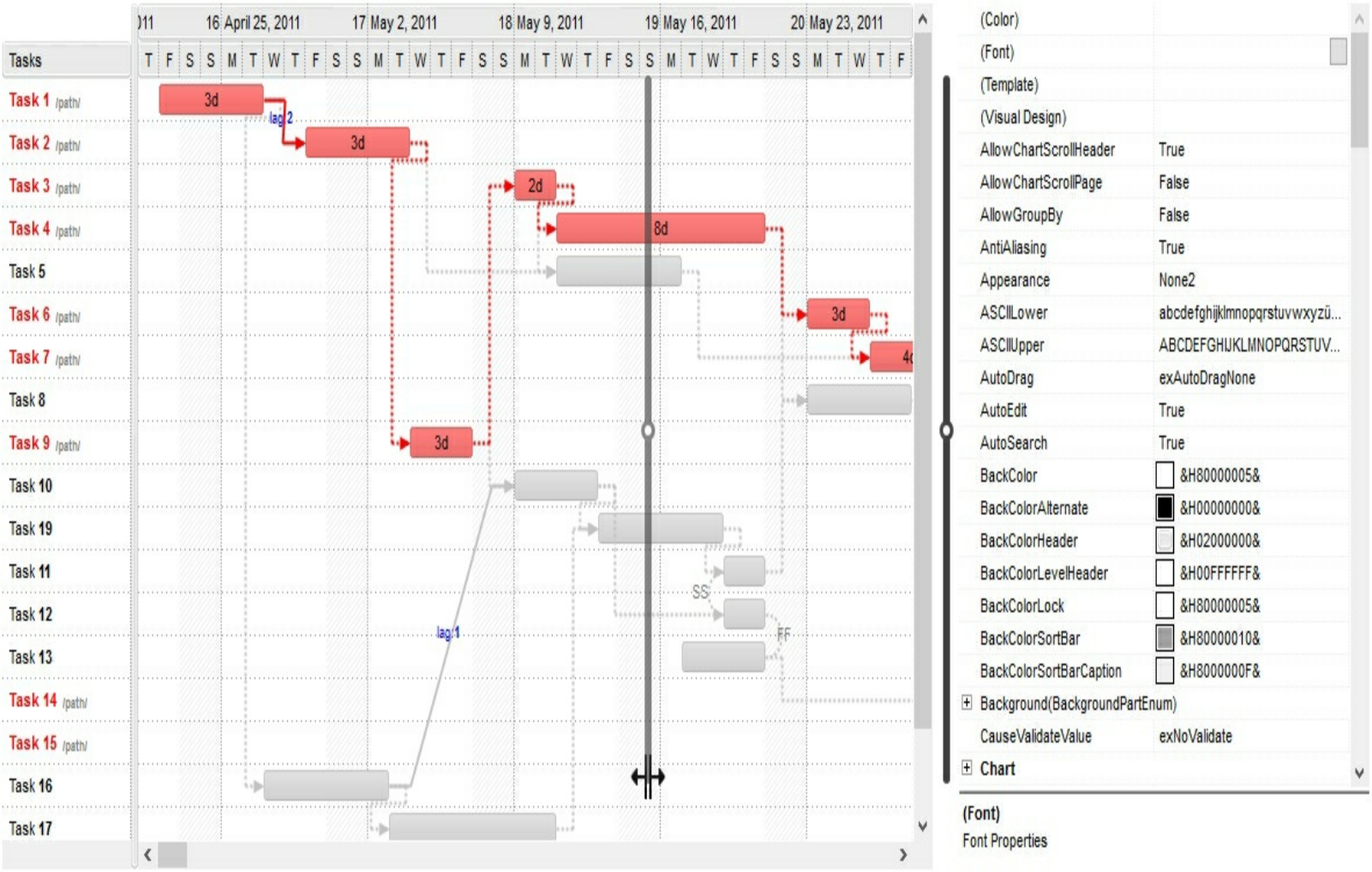
ExSplitBar

The Exontrol's eXSplitBar component, helps you to automatically resizes the left/right or top/bottom components, while user drags the split bar to a new position. The eXSplitBar control provides automatic resizing /positioning of controls/windows on your form/dialog/window. When the Mode property is set to exSplitBarHorz, the control resizes any controls that lie above or below it, and when the Mode is set to exSplitBarVert, it resizes controls that lie to its left or right. The difference between the eXSplitBar control and other components of the same type, is that the eXSplitBar control works for any programming environments such as Microsoft Office (Microsoft Access, Microsoft Excel, Microsoft Word), Visual Basic, Visual Fox Pro, /NET Framework, Delphi, C++, C++ Builder, dBASE Plus and more. As usual, there are no dependencies to MFC, VB, VCL, or anything else.

Features include:

- Easy to use, highly customizable
- Skinnable Interface support (ability to apply a skin to any background part)
- Horizontal, Vertical Mode support
- Requires absolutely no code
- Ability to specify the split bar's limits
- Ability to move controls as you move the split bar, not just when drop event occurs
- Ability to hide/shows controls when the split bar is close to the limit
- DragStart, Drag and DragEnd events support
- Ability to specify the split bar's limits
- Ability to specify the name of properties like: Left/Top, Width/Height, Visible, to use on non-standard containers



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 AppearanceEnum

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

Name	Value	Description
None2	0	The source has no borders.
Flat	1	Flat border
Sunken	2	Sunken border
Raised	3	Raised border
Etched	4	Etched border
Bump	5	Bump border

constants BackgroundPartEnum

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

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

constants PictureDisplayEnum

Specifies how a picture object is displayed.

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

constants SplitBarModeEnum

The SplitBarModeEnum type defines the type of split bar. The Mode property retrieves or sets a value that indicates the split bar's mode. The SplitBarModeEnum type supports the following values:

Name	Value	Description
exSplitBarAuto	0	By default, the Mode property exSplitBarAuto, which indicates that the split bar's mode is determines by its size as: • if the width of the split bar is greater or equal than its height, the Mode property is exSplitBarVert • if the width of the split bar is less than its height, the Mode property is exSplitBarHorz
exSplitBarHorz	1	If the control's Mode property is exSplitBarHorz, the split bar resizes any controls that lie above or below it.
exSplitBarVert	2	If the control's Mode property is exSplitBarVert, the split bar resizes controls that lie to its left or right.

Appearance object

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

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

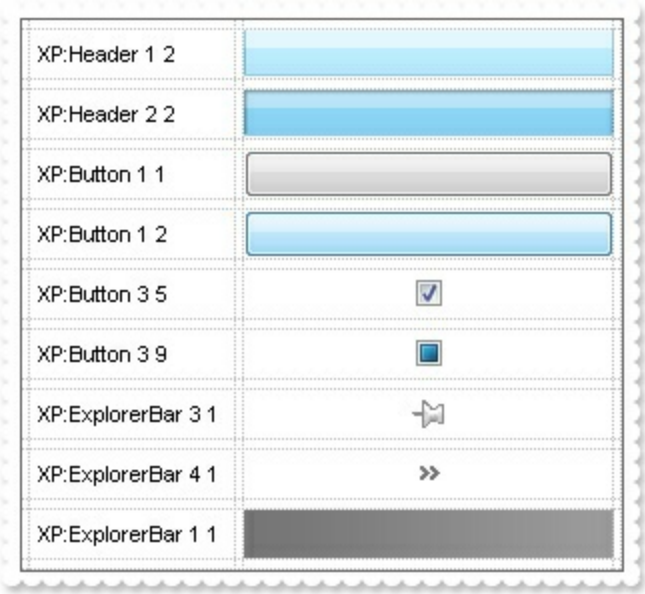
method Appearance.Add (ID as Long, Skin as Variant)

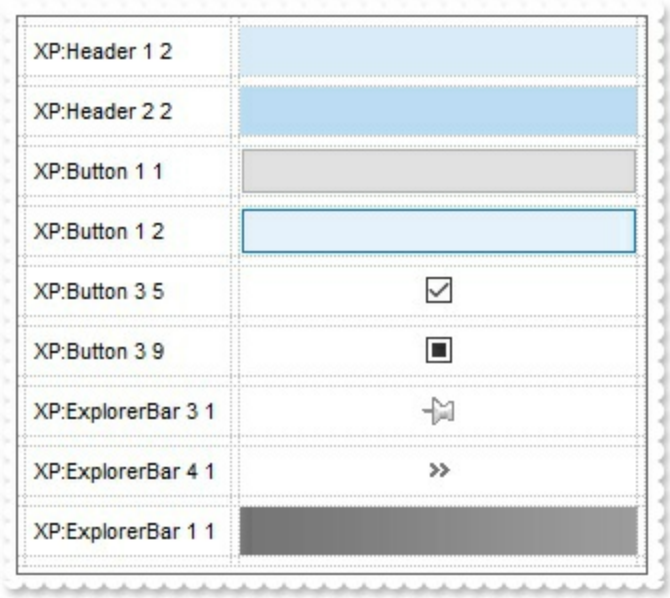
Adds or replaces a skin object to the control.

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

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

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



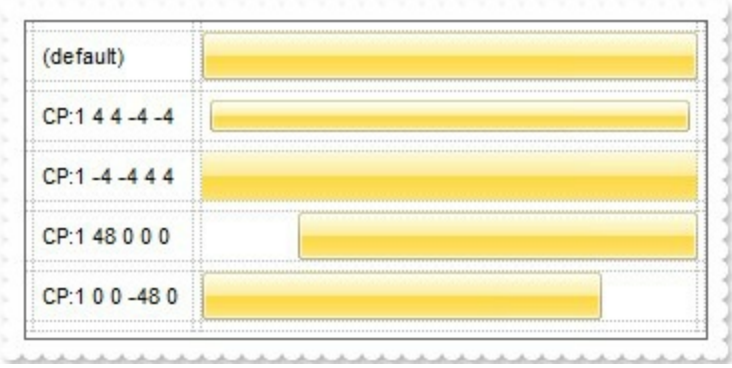


Skin as Variant

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

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

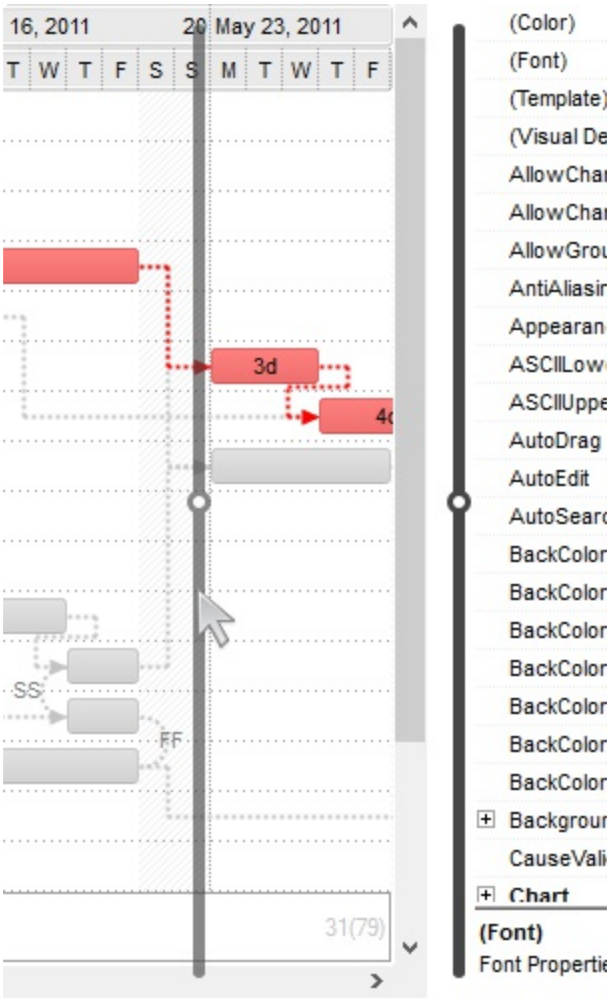
The following screen shot shows the same EBN being displayed, using different CP: options:



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

Use the Add method to add or replace skins to the control. The skin method, in it's simplest form, uses a single graphic file (*.ebn) assigned to a part of the control, when the "XP:" prefix is not specified in the Skin parameter (available for Windows XP systems). 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 do multiple changes to the control. Use the Refresh method to refresh the control.

The following screen shot shows the split bar with an EBN object:



The following screen shot shows the split bar with a solid color:

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_CHECKE 5 CBS_CHECKEDH CBS_CHECKEDPR CBS_CHECKEDDIS CBS_MIXEDNORM CBS_MIXEDHOT = CBS_MIXEDPRESS CBS_MIXEDDISAB
	BP_GROUPBOX = 4	GBS_NORMAL = 1 GBS_DISABLED =
	BP_PUSHBUTTON = 1	PBS_NORMAL = 1 = 2 PBS_PRESSED

		PBS_DISABLED =
		PBS_DEFAULTED =
		RBS_UNCHECKED
		1 RBS_UNCHECKE
		RBS_UNCHECKED
		= 3
	BP_RADIOBUTTON = 2	RBS_UNCHECKED
		= 4 RBS_CHECKE
		5 RBS_CHECKEDH
		RBS_CHECKEDPR
		RBS_CHECKEDDIS
	BP_USERBUTTON = 5	
CLOCK	CLP_TIME = 1	CLS_NORMAL = 1
		CBXS_NORMAL =
COMBOBOX	CP_DROPDOWNBUTTON = 1	CBXS_HOT = 2
		CBXS_PRESSED =
		CBXS_DISABLED =
EDIT	EP_CARET = 2	
		ETS_NORMAL = 1
		2 ETS_SELECTED
	EP_EDITTEXT = 1	ETS_DISABLED =
		ETS_FOCUSED = 5
		ETS_READONLY =
		ETS_ASSIST = 7
EXPLORERBAR	EBP_HEADERBACKGROUND = 1	
		EBHC_NORMAL =
	EBP_HEADERCLOSE = 2	EBHC_HOT = 2
		EBHC_PRESSED =
		EBHP_NORMAL =
		EBHP_HOT = 2
		EBHP_PRESSED =
	EBP_HEADERPIN = 3	EBHP_SELECTEDN
		4 EBHP_SELECTE
		EBHP_SELECTEDF
		6
	EBP_IEBARMENU = 4	EBM_NORMAL = 1
		= 2 EBM_PRESSE
	EBP_NORMALGROUPBACKGROUND = 5	
		EBNGC_NORMAL
	EBP_NORMALGROUPCOLLAPSE = 6	EBNGC_HOT = 2
		EBNGC_PRESSED

		EBNGE_NORMAL =
	EBP_NORMALGROUPEXPAND = 7	EBNGE_HOT = 2 EBNGE_PRESSED
	EBP_NORMALGROUPHEAD = 8	
	EBP_SPECIALGROUPBACKGROUND = 9	
		EBSGC_NORMAL =
	EBP_SPECIALGROUPCOLLAPSE = 10	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 H 2 HIS_PRESSED =
	HP_HEADERITEMLEFT = 2	HILS_NORMAL = 1 = 2 HILS_PRESSEI
	HP_HEADERITEMRIGHT = 3	HIRS_NORMAL = 1 = 2 HIRS_PRESSE
	HP_HEADERSORTARROW = 4	HSAS_SORTEDUP HSAS_SORTEDDC
LISTVIEW	LVP_EMPTYTEXT = 5	
	LVP_LISTDETAIL = 3	
	LVP_LISTGROUP = 2	
	LVP_LISTITEM = 1	LIS_NORMAL = 1 L 2 LIS_SELECTED = LIS_DISABLED = 4 LIS_SELECTEDNO 5
	LVP_LISTSORTEDDETAIL = 4	
MENU	MP_MENUBARDROPDOWN = 4	MS_NORMAL = 1 MS_SELECTED = 2 MS_DEMOTED = 3
	MP_MENUBARITEM = 3	MS_NORMAL = 1 MS_SELECTED = 2 MS_DEMOTED = 3
	MP_CHEVRON = 5	MS_NORMAL = 1 MS_SELECTED = 2 MS_DEMOTED = 3
		MS_NORMAL = 1 MS_SELECTED = 2

	MP_MENUDROPDOWN = 2	MS_DEMOTED = 3
	MP_MENUITEM = 1	MS_NORMAL = 1 MS_SELECTED = 2 MS_DEMOTED = 3
	MP_SEPARATOR = 6	MS_NORMAL = 1 MS_SELECTED = 2 MS_DEMOTED = 3
MENUBAND	MDP_NEWAPPBUTTON = 1	MDS_NORMAL = 1 = 2 MDS_PRESSE MDS_DISABLED = MDS_CHECKED = MDS_HOTCHECKE
	MDP_SEPERATOR = 2	
PAGE	PGRP_DOWN = 2	DNS_NORMAL = 1 = 2 DNS_PRESSED DNS_DISABLED =
	PGRP_DOWNHORZ = 4	DNHZS_NORMAL = DNHZS_HOT = 2 DNHZS_PRESSED DNHZS_DISABLED
	PGRP_UP = 1	UPS_NORMAL = 1 = 2 UPS_PRESSED UPS_DISABLED =
	PGRP_UPHORZ = 3	UPHZS_NORMAL = UPHZS_HOT = 2 UPHZS_PRESSED UPHZS_DISABLED
PROGRESS	PP_BAR = 1 PP_BARVERT = 2 PP_CHUNK = 3 PP_CHUNKVERT = 4	
REBAR	RP_BAND = 3	
	RP_CHEVRON = 4	CHEVS_NORMAL = CHEVS_HOT = 2 CHEVS_PRESSED
	RP_CHEVRONVERT = 5	
	RP_GRIPPER = 1	
	RP_GRIPPERVERT = 2	

SCROLLBAR	SBP_ARROWBTN = 1	ABS_DOWNDISAB ABS_DOWNHOT, ABS_DOWNNORM ABS_DOWNPRESS ABS_UPDISABLED ABS_UPHOT, ABS_UPNORMAL, ABS_UPPRESSED, ABS_LEFTDISABLI ABS_LEFTHOT, ABS_LEFTNORMA ABS_LEFTPRESSE ABS_RIGHTDISAB ABS_RIGHTHOT, ABS_RIGHTNORM ABS_RIGHTPRESS
	SBP_GRIPPERHORZ = 8 SBP_GRIPPERVERT = 9	
	SBP_LOWERTRACKHORZ = 4	SCRBS_NORMAL = SCRBS_HOT = 2 SCRBS_PRESSED SCRBS_DISABLED
	SBP_LOWERTRACKVERT = 6	SCRBS_NORMAL = SCRBS_HOT = 2 SCRBS_PRESSED SCRBS_DISABLED
	SBP_THUMBBTNHORZ = 2	SCRBS_NORMAL = SCRBS_HOT = 2 SCRBS_PRESSED SCRBS_DISABLED
	SBP_THUMBBTNVERT = 3	SCRBS_NORMAL = SCRBS_HOT = 2 SCRBS_PRESSED SCRBS_DISABLED
	SBP_UPPERTRACKHORZ = 5	SCRBS_NORMAL = SCRBS_HOT = 2 SCRBS_PRESSED SCRBS_DISABLED
	SBP_UPPERTRACKVERT = 7	SCRBS_NORMAL = SCRBS_HOT = 2 SCRBS_PRESSED SCRBS_DISABLED

	SBP_SIZEBOX = 10	SZB_RIGHTALIGN SZB_LEFTALIGN =
SPIN	SPNP_DOWN = 2	DNS_NORMAL = 1 = 2 DNS_PRESSED DNS_DISABLED =
	SPNP_DOWNHORZ = 4	DNHZS_NORMAL = DNHZS_HOT = 2 DNHZS_PRESSED DNHZS_DISABLED
	SPNP_UP = 1	UPS_NORMAL = 1 = 2 UPS_PRESSED UPS_DISABLED =
	SPNP_UPHORZ = 3	UPHZS_NORMAL = UPHZS_HOT = 2 UPHZS_PRESSED UPHZS_DISABLED
STARTPANEL	SPP_LOGOFF = 8	
	SPP_LOGOFFBUTTONS = 9	SPLS_NORMAL = SPLS_HOT = 2 SPLS_PRESSED =
	SPP_MOREPROGRAMS = 2	
	SPP_MOREPROGRAMSARROW = 3	SPS_NORMAL = 1 = 2 SPS_PRESSED
	SPP_PLACESLIST = 6	
	SPP_PLACESLISTSEPARATOR = 7	
	SPP_PREVIEW = 11	
	SPP_PROGLIST = 4	
	SPP_PROGLISTSEPARATOR = 5	
	SPP_USERPANE = 1	
	SPP_USERPICTURE = 10	
STATUS	SP_GRIPPER = 3	
	SP_PANE = 1	
	SP_GRIPPERPANE = 2	
TAB	TABP_BODY = 10	
	TABP_PANE = 9	
	TABP_TABITEM = 1	TIS_NORMAL = 1 2 TIS_SELECTED = TIS_DISABLED = 4 TIS_FOCUSED = 5

	TABP_TABITEMBOTHEDGE = 4	TIBES_NORMAL = TIBES_HOT = 2 TIBES_SELECTED TIBES_DISABLED TIBES_FOCUSED =
	TABP_TABITEMLEFTEDGE = 2	TILES_NORMAL = TILES_HOT = 2 TILES_SELECTED TILES_DISABLED = TILES_FOCUSED =
	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_SELECTE TTILES_DISABLED TTILES_FOCUSED
	TABP_TOPTABITEMRIGHTEDGE = 7	TTIRES_NORMAL TTIRES_HOT = 2 TTIRES_SELECTE TTIRES_DISABLED TTIRES_FOCUSED
TASKBAND	TDP_GROUPCOUNT = 1	
	TDP_FLASHBUTTON = 2	
	TDP_FLASHBUTTONGROUPMENU = 3	
TASKBAR	TBP_BACKGROUNDBOTTOM = 1	
	TBP_BACKGROUNDLEFT = 4	
	TBP_BACKGROUNDRIGHT = 2	
	TBP_BACKGROUNDTOP = 3	

	TBP_SIZINGBARBOTTOM = 5	
	TBP_SIZINGBARBOTTOMLEFT = 8	
	TBP_SIZINGBARRIGHT = 6	
	TBP_SIZINGBARTOP = 7	
TOOLBAR	TP_BUTTON = 1	TS_NORMAL = 1 T TS_PRESSED = 3 TS_DISABLED = 4 TS_CHECKED = 5 TS_HOTCHECKED
	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 = 1 TTSS_LINK = 2
	TTP_STANDARDTITLE = 2	TTSS_NORMAL = 1 TTSS_LINK = 2
TRACKBAR	TKP_THUMB = 3	TUS_NORMAL = 1 2 TUS_PRESSED = TUS_FOCUSED = 4 TUS_DISABLED =
	TKP_THUMBBOTTOM = 4	TUBS_NORMAL = TUBS_HOT = 2 TUBS_PRESSED = TUBS_FOCUSED = TUBS_DISABLED =
	TKP_THUMBLEFT = 7	TUVLS_NORMAL = TUVLS_HOT = 2 TUVLS_PRESSED TUVLS_FOCUSED TUVLS_DISABLED
	TKP_THUMBRIGHT = 8	TUVRS_NORMAL = TUVRS_HOT = 2 TUVRS_PRESSED TUVRS_FOCUSED TUVRS_DISABLED
	TKP_THUMBTOP = 5	TUTS_NORMAL = 1 TUTS_HOT = 2 TUTS_PRESSED = TUTS_FOCUSED = TUTS_DISABLED =
	TKP_THUMBVERT = 6	TUVS_NORMAL = TUVS_HOT = 2 TUVS_PRESSED = TUVS_FOCUSED = TUVS_DISABLED =
	TKP_TICS = 9	TSS_NORMAL = 1
	TKP_TICSVERT = 10	TSVS_NORMAL =
	TKP_TRACK = 1	TRS_NORMAL = 1
	TKP_TRACKVERT = 2	TRVS_NORMAL =
TRAYNOTIFY	TNP_ANIMBACKGROUND = 2	
	TNP_BACKGROUND = 1	

TREEVIEW	TVP_BRANCH = 3	
	TVP_GLYPH = 2	GLPS_CLOSED = GLPS_OPENED =
	TVP_TREEITEM = 1	TREIS_NORMAL = TREIS_HOT = 2 TREIS_SELECTED TREIS_DISABLED TREIS_SELECTED = 5
WINDOW	WP_CAPTION = 1	CS_ACTIVE = 1 CS = 2 CS_DISABLED
	WP_CAPTIONSIZINGTEMPLATE = 30	
	WP_CLOSEBUTTON = 18	CBS_NORMAL = 1 = 2 CBS_PUSHED CBS_DISABLED =
	WP_DIALOG = 29	
	WP_FRAMEBOTTOM = 9	FS_ACTIVE = 1 FS = 2
	WP_FRAMEBOTTOMSIZINGTEMPLATE = 36	
	WP_FRAMELEFT = 7	FS_ACTIVE = 1 FS = 2
	WP_FRAMELEFTSIZINGTEMPLATE = 32	
	WP_FRAMERIGHT = 8	FS_ACTIVE = 1 FS = 2
	WP_FRAMERIGHTSIZINGTEMPLATE = 34	
	WP_HELPBUTTON = 23	HBS_NORMAL = 1 = 2 HBS_PUSHED HBS_DISABLED =
	WP_HORZSCROLL = 25	HSS_NORMAL = 1 = 2 HSS_PUSHED HSS_DISABLED =
	WP_HORZTHUMB = 26	HTS_NORMAL = 1 2 HTS_PUSHED = 3 HTS_DISABLED = 4
	WP_MAX_BUTTON	MAXBS_NORMAL MAXBS_HOT = 2 MAXBS_PUSHED = MAXBS_DISABLED
	WP_MAXCAPTION = 5	MXCS_ACTIVE = 1 MXCS_INACTIVE =

	MXCS_DISABLED
WP_MDICLOSEBUTTON = 20	CBS_NORMAL = 1 = 2 CBS_PUSHED CBS_DISABLED =
WP_MDIHELPBUTTON = 24	HBS_NORMAL = 1 = 2 HBS_PUSHED HBS_DISABLED =
WP_MDIMINBUTTON = 16	MINBS_NORMAL = MINBS_HOT = 2 MINBS_PUSHED = MINBS_DISABLED
WP_MDIRESTOREBUTTON = 22	RBS_NORMAL = 1 = 2 RBS_PUSHED RBS_DISABLED =
WP_MDISYSBUTTON = 14	SBS_NORMAL = 1 = 2 SBS_PUSHED SBS_DISABLED =
WP_MINBUTTON = 15	MINBS_NORMAL = MINBS_HOT = 2 MINBS_PUSHED = MINBS_DISABLED
WP_MINCAPTION = 3	MNCS_ACTIVE = 1 MNCS_INACTIVE = MNCS_DISABLED
WP_RESTOREBUTTON = 21	RBS_NORMAL = 1 = 2 RBS_PUSHED RBS_DISABLED =
WP_SMALLCAPTION = 2	CS_ACTIVE = 1 CS = 2 CS_DISABLED
WP_SMALLCAPTIONSIZINGTEMPLATE = 31	
WP_SMALLCLOSEBUTTON = 19	CBS_NORMAL = 1 = 2 CBS_PUSHED CBS_DISABLED =
WP_SMALLFRAMEBOTTOM = 12	FS_ACTIVE = 1 FS = 2
WP_SMALLFRAMEBOTTOMSIZINGTEMPLATE = 37	
WP_SMALLFRAMELEFT = 10	FS_ACTIVE = 1 FS = 2
WP_SMALLFRAMELEFTSIZINGTEMPLATE =	

33	
WP_SMALLFRAMERIGHT = 11	FS_ACTIVE = 1 FS = 2
WP_SMALLFRAMERIGHTSIZINGTEMPLATE = 35	
WP_SMALLHELPBUTTON	HBS_NORMAL = 1 = 2 HBS_PUSHED HBS_DISABLED =
WP_SMALLMAXBUTTON	MAXBS_NORMAL MAXBS_HOT = 2 MAXBS_PUSHED = MAXBS_DISABLED
WP_SMALLMAXCAPTION = 6	MXCS_ACTIVE = 1 MXCS_INACTIVE = MXCS_DISABLED
WP_SMALLMINCAPTION = 4	MNCS_ACTIVE = 1 MNCS_INACTIVE = MNCS_DISABLED
WP_SMALLRESTOREBUTTON	RBS_NORMAL = 1 = 2 RBS_PUSHED RBS_DISABLED =
WP_SMALLSYSBUTTON	SBS_NORMAL = 1 = 2 SBS_PUSHED SBS_DISABLED =
WP_SYSBUTTON = 13	SBS_NORMAL = 1 = 2 SBS_PUSHED SBS_DISABLED =
WP_VERTSCROLL = 27	VSS_NORMAL = 1 = 2 VSS_PUSHED VSS_DISABLED =
WP_VERTTHUMB = 28	VTS_NORMAL = 1 2 VTS_PUSHED = 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
```

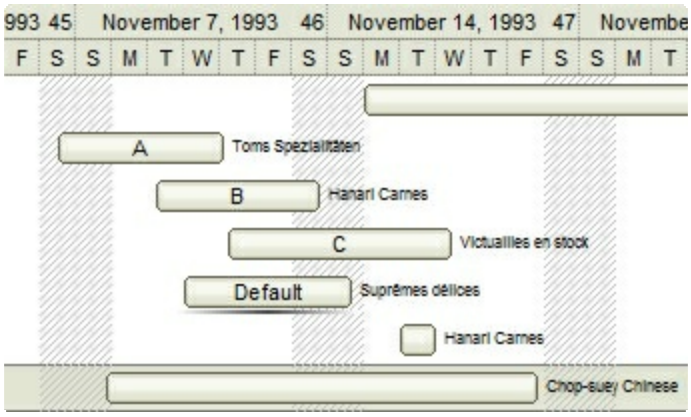
In the following screen shot the following objects displays the current EBN with a different color:

- "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

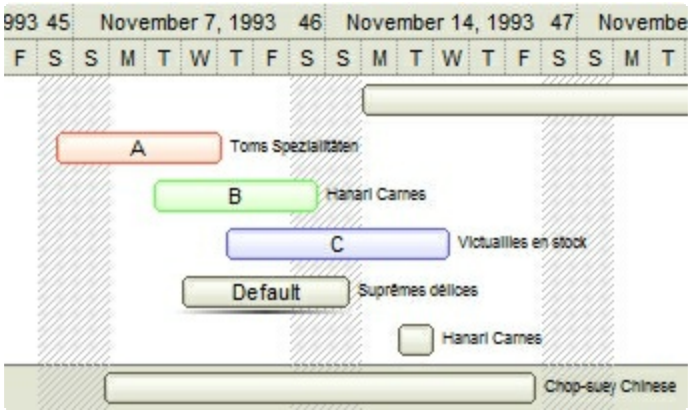
- "C" in Blue (RGB(0,0,255), for instance the bar's property exBarColor is 0x1FF0000
- "Default", no color is specified, for instance the bar's property exBarColor is 0x1000000

The RenderType property could be one of the following:

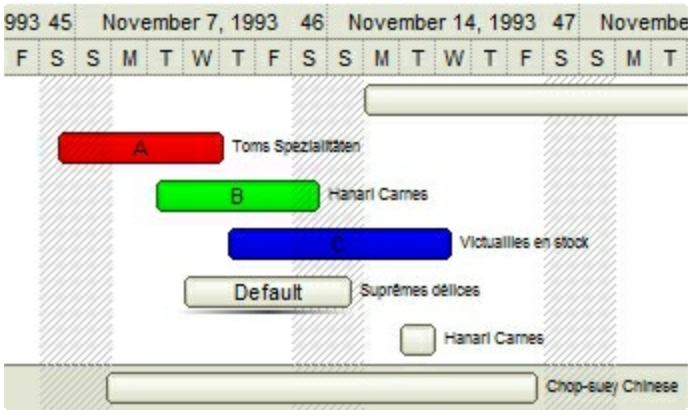
- ***-3**, no color is applied*. For instance, the BackColorHeader = &H1FF0000 is displayed as would be .BackColorHeader = &H1000000, so the 0xFF0000 color (Blue color) is ignored. You can use this option to allow the control displays the EBN colors or not.



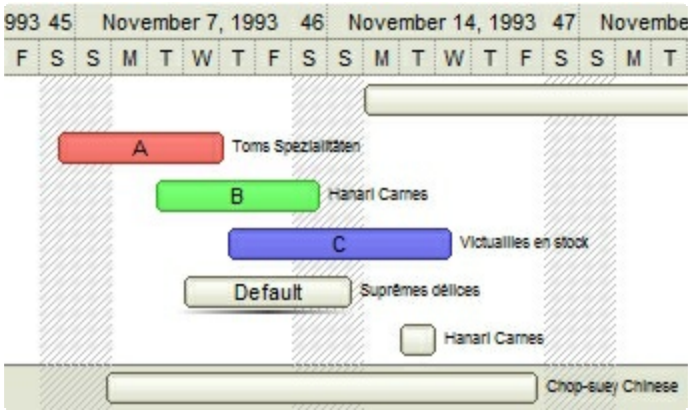
- ***-2**, OR-color scheme*. The color to be applied on the part of the control is a OR bit combination between the original EBN color and the specified color. For instance, the BackColorHeader = &H1FF0000, applies the OR bit for the entire Blue channel, or in other words, it applies a less Blue to the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

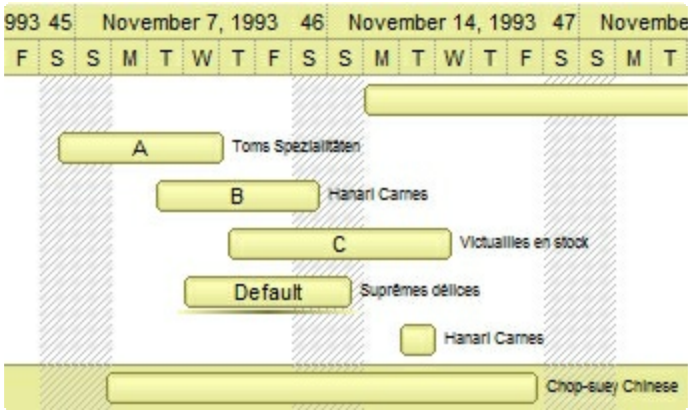


- ***0**, default*, the specified color is applied to the EBN. For instance, the BackColorHeader = &H1FF0000, applies a Blue color to the object. This option could be used to specify any color for the part of the components, that support EBN objects, not only solid colors.

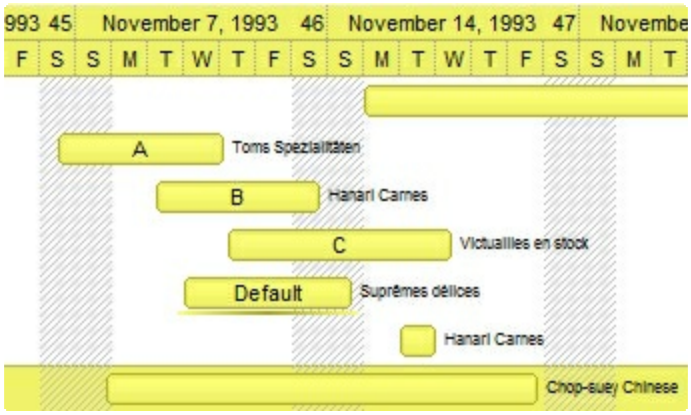


- **0xAABBGGRR**, where the AA a value between 0 to 255, which indicates the transparency, and RR, GG, BB the red, green and blue values. This option applies the same color to all parts that displays EBN objects, whit ignoring any specified color in the color property. For instance, the RenderType on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

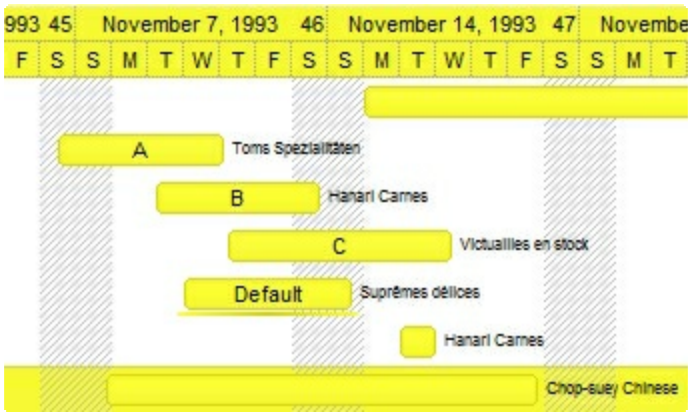
*The following picture shows the control with the RenderType property on 0x**40**00FFFF (25% Yellow, 0x40 or 64 in decimal is 25% from 256):*



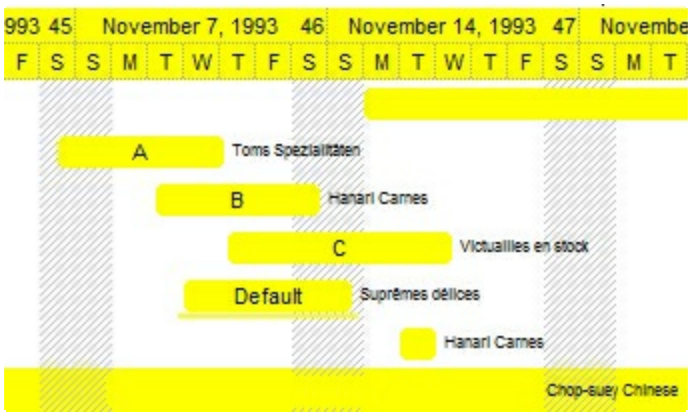
*The following picture shows the control with the RenderType property on 0x**80**00FFFF (50% Yellow, 0x80 or 128 in decimal is 50% from 256):*



*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

SplitBar 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: {9F28FDED-5EBC-4E9A-A596-C3813C966A0C}. The object's program identifier is: "Exontrol.SplitBar". The /COM object module is: "ExSplitBar.dll"

The Exontrol's eXSplitBar component, helps you to automatically resizes the left/right or top/bottom components, while user drags the split bar to a new position. The eXSplitBar control provides automatic resizing /positioning of controls/windows on your form/dialog/window. When the Mode property is set to exSplitBarHorz, the control resizes any controls that lie above or below it, and when the Mode is set to exSplitBarVert, it resizes controls that lie to its left or right. The difference between the eXSplitBar control and other components of the same type, is that the eXSplitBar control works for any programming environments such as Microsoft Office (Microsoft Access, Microsoft Excel, Microsoft Word), Visual Basic, Visual Fox Pro, /NET Framework, Delphi, C++, and more. The eXSplitBar supports the following properties and methods:

Name	Description
AddObjectLT	Adds a new object to be updated in the left/top part of the split bar.
AddObjectRB	Adds a new object to be updated in the right/bottom part of the split bar.
Appearance	Retrieves or sets the control's appearance.
AttachTemplate	Attaches a script to the current object, including the events, from a string, file, a safe array of bytes.
BackColor	Specifies the control's background color.
Background	Returns or sets a value that indicates the background color for parts in the control.
BeginUpdate	Maintains performance when multiple changes are performed 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.
Cursor	Gets or sets the cursor that is displayed when the mouse pointer hovers the control.
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.
ExecuteTemplate	Executes a template and returns the result.
ExtendedContainerWnd	Specifies the list of window class names of parents added by the extended control.
ExtendedHeight	Specifies a list of property names separated by comma character, that indicates the Height property of the extended control. The Height property of an extended control gets or sets the control's height.
ExtendedLeft	Specifies a list of property names separated by comma character, that indicates the Left property of the extended control. The Left property of an extended control gets or sets the distance, between the left edge of the object and the left edge of its containe?Â?`
ExtendedName	Specifies a list of property names separated by comma character, that indicates the Name property of the extended control. The Name property of an extended control specifies the name of the object within the container.
ExtendedObject	Specifies a list of property names separated by comma character, that indicates the Object property of the extended control. The Object property of an extended control returns the original /hosted object.
ExtendedTop	Specifies a list of property names separated by comma character, that indicates the Top property of the extended control. The Top property of an extended control gets or sets the distance, between the top edge of the object and the top edge of its container.
ExtendedVisible	Specifies a list of property names separated by comma character, that indicates the Visible property of the extended control. The Visible property of an extended control shows or hides object.
ExtendedWidth	Specifies a list of property names separated by comma character, that indicates the Width property of the extended control. The Width property of an extended control gets or sets the control's width.
Font	Retrieves or sets the control's font.
ForeColor	Specifies the control's foreground color.

HideOnLimit	Gets or sets a value that indicates whether the splitting objects are hidden when the split bar is closed to its limit.
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..
LimitLT	Specifies the expression that determines the limit to drag the splitter to left/top side of its container.
LimitRB	Specifies the expression that determines the limit to drag the splitter to right/bottom side of its container.
Max	Indicates rightmost/bottommost position the split bar can be moved.
Min	Indicates leftmost/topmost position the split bar can be moved.
Mode	Retrieves or sets a value that indicates the split bar's mode.
MoveOnDrop	Gets or sets a value that indicates whether the splitting objects (including the split bar itself) are moved once the user ends dragging the split bar, or contiguously while dragging it.
MoveTo	Moves the split bar to the specified position.
ObjectsIN	Specifies a list of controls that are child of other controls (prevents changing the left/top part of the control while it is relative to a different parent).
ObjectsLT	Indicates the object to be updated in the left/top part of the split bar.
ObjectsRB	Indicates the object to be updated in the right/bottom part of the split bar.
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.

ShowImageList	Specifies whether the control's image list window is visible or hidden.
SplitBackColor	Specifies the splitter's background color.
SplitHotBackColor	Specifies the splitter's background color, while the cursor is hovering it.
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.
ToolTipText	Specifies the control's tooltip text.
ToolTipTitle	Specifies the title of the control's tooltip.
ToolTipWidth	Specifies a value that indicates the width of the tooltip window, in pixels.
Version	Retrieves the control's version.
VisualAppearance	Retrieves the control's appearance.

method SplitBar.AddObjectLT (newVal as Variant)

Adds a new object to be updated in the left/top part of the split bar.

Type	Description
newVal as Variant	A Variant expression that could be one of the following: • String expression that indicates the name of the component. For instance: AddObjectLT("Command1") • A String expression that represents a number, which is the handle of the window. For instance, AddObjectLT(CStr(Grid1.hWnd)) As string, the AddObjectLT method can add multiple values at the same time, if passing a string, with values as explained above, separated by comma character. For instance: AddObjectLT("Command1,Command2," + CStr(Grid1.hWnd)) • A numeric expression that indicates the handle of the window. For instance, AddObjectLT(Grid1.hWnd) • An IUnknown interface, that indicates a reference to the object to be anchored (/COM only). For instance, AddObjectLT(GetDlgItem(IDC_SPLITBAR1)->GetControlUnknown()), VC++ • A IDispatch interface, that indicates a reference to the object to be anchored (/COM only). For instance, AddObjectLT(SplitBar1.DefaultInterface), Delphi • An object of Control (System.Windows.Forms) type that specifies the control (/NET assembly). For instance, AddObjectLT(exgrid1), C# A safe array of a VARIANT type, with any value explained above. For instance, AddObjectLT(Array("Command1", Command2, Grid1.hWnd)), VB/NET

The AddObjectLT method adds at runtime, a new object to be updated in the left/top part of the split bar. The ObjectsLT property defines the controls associated with the left/top side of the split bar at design mode. The Mode property specifies whether the split bar moves objects horizontally or vertically. When the Mode property is set to exSplitBarHorz, the control resizes any controls that lie above or below it, and when the Mode is set to

exSplitBarVert, it resizes controls that lie to its left or right. The ObjectsRB property defines the objects to be updated on the right/bottom side of the split bar. Setting the ObjectsLT property on "" (empty string), releases any control/object that has been previously anchored to the slit bar, including the objects being added with the AddObjectLT method, or the split bar has nothing attached to its left/top side. The LimitLT property specifies the expression that determines the limit to drag the splitter to left/top side of its container.

By default, if a control/component/object is contained in

- both ObjectsLT / AddObjectLT and ObjectsRB / AddObjectRB, the control/component/object will be moved (not sized), when the split bar moves
- else the control/component/object will be moved and sized accordingly with the side of the split bar it is anchored.

In

- C++ Builder
- C# for /COM on /NET Framework
- Delphi
- Visual Basic for /COM on /NET Framework
- Visual C++

you need to use the AddObjectLT and AddObjectRB methods as in the following samples.

C++ Builder :

```
SplitBar1->AddObjectLT(TVariant(Button1->Handle));
SplitBar1->AddObjectRB(TVariant(Button2->Handle));
SplitBar1->AddObjectRB(TVariant(SplitBar2->DefaultInterface));
SplitBar1->AddObjectRB(TVariant(Button3->Handle));
```

C# for /COM on /NET Framework :

```
axSplitBar1.AddObjectLT(button1);
axSplitBar1.AddObjectRB(button2);
axSplitBar1.AddObjectRB(axSplitBar2);
axSplitBar1.AddObjectRB(button3);
```

Delphi :

```
with SplitBar1 do
begin
  AddObjectLT(Button1.Handle);
```

```
    AddObjectRB(Button2.Handle);
    AddObjectRB(SplitBar2.DefaultInterface);
    AddObjectRB(Button3.Handle);
end
```

Visual Basic for /COM on /NET Framework:

```
With AxSplitBar1
    .AddObjectLT(Button1)
    .AddObjectRB(Button2)
    .AddObjectRB(AxSplitBar2)
    .AddObjectRB(Button3)
End With
```

Visual C++:

```
EXSPLITBARLib::ISplitBarPtr spSplitBar1 = GetDlgItem(IDC_SPLITBAR1)-
>GetControlUnknown();
spSplitBar1->AddObjectLT( (long)::GetDlgItem( m_hWnd, IDC_BUTTON1 ) );
spSplitBar1->AddObjectRB( (long)::GetDlgItem( m_hWnd, IDC_BUTTON2 ) );
spSplitBar1->AddObjectRB( GetDlgItem(IDC_SPLITBAR2)->GetControlUnknown() );
spSplitBar1->AddObjectRB( (long)::GetDlgItem( m_hWnd, IDC_BUTTON3 ) );
```

method SplitBar.AddObjectRB (newVal as Variant)

Adds a new object to be updated in the right/bttom part of the split bar.

Type	Description
newVal as Variant	A Variant expression that could be one of the following: • String expression that indicates the name of the component. For instance: AddObjectRB("Command1") • A String expression that represents a number, which is the handle of the window. For instance, AddObjectRB(CStr(Grid1.hWnd)) As string, the AddObjectRB method can add multiple values at the same time, if passing a string, with values as explained above, separated by comma character. For instance: AddObjectRB("Command1,Command2," + CStr(Grid1.hWnd)) • A numeric expression that indicates the handle of the window. For instance, AddObjectRB(Grid1.hWnd) • An IUnknown interface, that indicates a reference to the object to be anchored (/COM only). For instance, AddObjectRB(GetDlgItem(IDC_SPLITBAR1)->GetControlUnknown()), VC++ • A IDispatch interface, that indicates a reference to the object to be anchored (/COM only). For instance, AddObjectRB(SplitBar1.DefaultInterface), Delphi • An object of Control (System.Windows.Forms) type that specifies the control (/NET assembly). For instance, AddObjectRB(exgrid1), C# A safe array of a VARIANT type, with any value explained above. For instance, AddObjectRB(Array("Command1", Command2, Grid1.hWnd)), VB/NET

The AddObjectRB method adds at runtime, a new object to be updated in the right/bottom part of the split bar. The ObjectsRB property defines the objects to be updated on the right/bottom side of the split bar. The ObjectsLT property defines the controls associated with the left/top side of the split bar at design mode. The Mode property specifies whether the split bar moves objects horizontally or vertically. When the Mode property is set to

exSplitBarHorz, the control resizes any controls that lie above or below it, and when the Mode is set to exSplitBarVert, it resizes controls that lie to its left or right. Setting the ObjectsRB property on "" (empty string), releases any control/object that has been previously anchored to the slit bar, including the objects being added with the AddObjectRB method, or the split bar has nothing attached to its left/top side. The LimitRB property specifies the expression that determines the limit to drag the splitter to right/bottom side of its container.

By default, if a control/component/object is contained in

- both ObjectsLT / AddObjectLT and ObjectsRB / AddObjectRB, the control/component/object will be moved (not sized), when the split bar moves
- else the control/component/object will be moved and sized accordingly with the side of the split bar it is anchored.

In

- C++ Builder
- C# for /COM on /NET Framework
- Delphi
- Visual Basic for /COM on /NET Framework
- Visual C++

you need to use the AddObjectLT and AddObjectRB methods as in the following samples.

C++ Builder :

```
SplitBar1->AddObjectLT(TVariant(Button1->Handle));
SplitBar1->AddObjectRB(TVariant(Button2->Handle));
SplitBar1->AddObjectRB(TVariant(SplitBar2->DefaultInterface));
SplitBar1->AddObjectRB(TVariant(Button3->Handle));
```

C# for /COM on /NET Framework :

```
axSplitBar1.AddObjectLT(button1);
axSplitBar1.AddObjectRB(button2);
axSplitBar1.AddObjectRB(axSplitBar2);
axSplitBar1.AddObjectRB(button3);
```

Delphi :

```
with SplitBar1 do
begin
```

```
    AddObjectLT(Button1.Handle);
    AddObjectRB(Button2.Handle);
    AddObjectRB(SplitBar2.DefaultInterface);
    AddObjectRB(Button3.Handle);
end
```

Visual Basic for /COM on /NET Framework:

```
With AxSplitBar1
    .AddObjectLT(Button1)
    .AddObjectRB(Button2)
    .AddObjectRB(AxSplitBar2)
    .AddObjectRB(Button3)
End With
```

Visual C++:

```
EXSPLITBARLib::ISplitBarPtr spSplitBar1 = GetDlgItem(IDC_SPLITBAR1)-
>GetControlUnknown();
spSplitBar1->AddObjectLT( (long)::GetDlgItem( m_hWnd, IDC_BUTTON1 ) );
spSplitBar1->AddObjectRB( (long)::GetDlgItem( m_hWnd, IDC_BUTTON2 ) );
spSplitBar1->AddObjectRB( GetDlgItem(IDC_SPLITBAR2)->GetControlUnknown() );
spSplitBar1->AddObjectRB( (long)::GetDlgItem( m_hWnd, IDC_BUTTON3 ) );
```

property SplitBar.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 list/hierarchy, scrollbars are always shown in the control's client area. The skin may contain transparent objects, and so you can define round corners. The normal.ebn file contains such of objects. Use the eXButton's Skin builder to view or change this file***

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

method SplitBar.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 SplitBar1_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 SplitBar.BackColor as Color

Specifies the control's background color.

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

The BackColor property specifies the control's background color. The ForeColor property specifies the control's foreground color. The SplitBarBackColor property defines the slit bar's visual appearance/background color. The SplitHotBackColor property defines the slit bar's visual appearance/background color, while cursor is hovering the split bar. The BorderWidth property sets or retrieves a value that indicates the border width of the control. The BorderHeight property sets or retrieves a value that indicates the border height of the control.

property SplitBar.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 BeginUpdate and EndUpdate methods to maintain performance while init the control. Use the Refresh method to refresh the control. The BackColor property specifies the control's background color. The SplitBarBackColor property defines the slit bar's visual appearance/background color. The SplitHotBackColor property defines the slit bar's visual appearance/background color, while cursor is hovering the split bar. The BorderWidth property sets or retrieves a value that indicates the border width of the control. The BorderHeight property sets or retrieves a value that indicates the border height of the control.

method SplitBar.BeginUpdate ()

Maintains performance when multiple changes are performed one at a time

Type	Description

This method prevents the control from painting until the EndUpdate method is called. The Refresh method refreshes the control.

property SplitBar.BorderHeight as Long

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

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

By default, the BorderHeight property is 2 pixels. The BorderHeight property sets or retrieves a value that indicates the border height of the control. The BorderWidth property sets or retrieves a value that indicates the border width of the control. The SplitHotBackColor property defines the slit bar's visual appearance/background color, while cursor is hovering the split bar. The SplitBarBackColor property defines the slit bar's visual appearance/background color. The BackColor property specifies the control's background color.

property SplitBar.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 2 pixels. The BorderWidth property sets or retrieves a value that indicates the border width of the control. The BorderHeight property sets or retrieves a value that indicates the border height of the control. The SplitHotBackColor property defines the slit bar's visual appearance/background color, while cursor is hovering the split bar. The SplitBarBackColor property defines the slit bar's visual appearance/background color. The BackColor property specifies the control's background color.

property SplitBar.Cursor as Variant

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

Type	Description
Variant	The VARIANT expression that could be: • A string expression that indicates a predefined value listed bellow • A string expression that indicates the path to a cursor file • A long expression that indicates the handle of the cursor.

By default, the Cursor property is "exDefault", which indicates that the split bar determines the shape of the cursor based on the split bar's mode. The Mode property specifies whether the split bar moves objects horizontally or vertically. When the Mode property is set to exSplitBarHorz, the control resizes any controls that lie above or below it, and when the Mode is set to exSplitBarVert, it resizes controls that lie to its left or right. Use the Cursor property to specify the cursor that control displays when mouse pointer hovers the split bar. The Cursor property has no effect if the split bar is disabled. The Enabled property specifies whether the control is enabled or disabled.

Here's the list of predefined values (string expressions):

- **"exDefault"** - (Default) Shape determined by the object.
- **"exArrow"** - Arrow.
- **"exCross"** - Cross (cross-hair pointer).
- **"exIBeam"** - I Beam.
- **"exIcon"** - Icon (small square within a square).
- **"exSize" -** Size (four-pointed arrow pointing north, south, east, and west).
- **"exSizeNESW"** - Size NE SW (double arrow pointing northeast and southwest).
- **"exSizeNS"** - Size N S (double arrow pointing north and south).
- **"exSizeNWSE"** - Size NW, SE.
- **"exSizeWE"** - Size W E (double arrow pointing west and east).
- **"exUpArrow"** - Up Arrow.
- **"exHourglass"** - Hourglass (wait).
- **"exNoDrop"** - No Drop.
- **"exArrowHourglass"** - Arrow and hourglass.
- **"exHelp"** - Arrow and question mark.
- **"exSizeAll"** - Size all.

property SplitBar.Enabled as Boolean

Enables or disables the control.

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

By default, the Enabled property is True. The Enabled property specifies whether the control is enabled or disabled. You can use the Enabled property, to disable splitting the control. You can disable dragging the split bar, if setting the Cancel parameter to True, during the DragStart event. The Cursor property specifies the shape of the cursor when the cursor is hovering the split bar. The control fires DragStart event when the user clicks the split bar (start dragging the split bar). The Drag event is fired contiguously while the split bar is dragging. The DragEnd event notifies your application that the user releases the split bar (ends dragging the split bar). The MoveTo method moves programmatically the split bar to specified position. The MoveOnDrop property specifies whether the objects to the left/top and right/bottom of the split bar are moved while dragging or just when the user drops the split bar. Use the Cursor property to specify the cursor that control displays when mouse pointer hovers the split bar.

method SplitBar.EndUpdate ()

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

Type	Description

Use BeginUpdate and EndUpdate statement each time when the control requires more changes. The Refresh method refreshes the control.

property SplitBar.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 SplitBar.ExecuteTemplate (Template as String)

Executes a template and returns the result.

Type	Description
Template as String	A Template string being executed

Return	Description
Variant	A Variant expression that indicates the result after executing the Template.

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

For instance, the following sample retrieves the control's background color:

```
Debug.Print SplitBar1.ExecuteTemplate("BackColor")
```

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

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

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

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

newer versions of the components.

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

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

The x-script may uses constant expressions as follow:

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

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

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

the class associated with a specified program identifier.

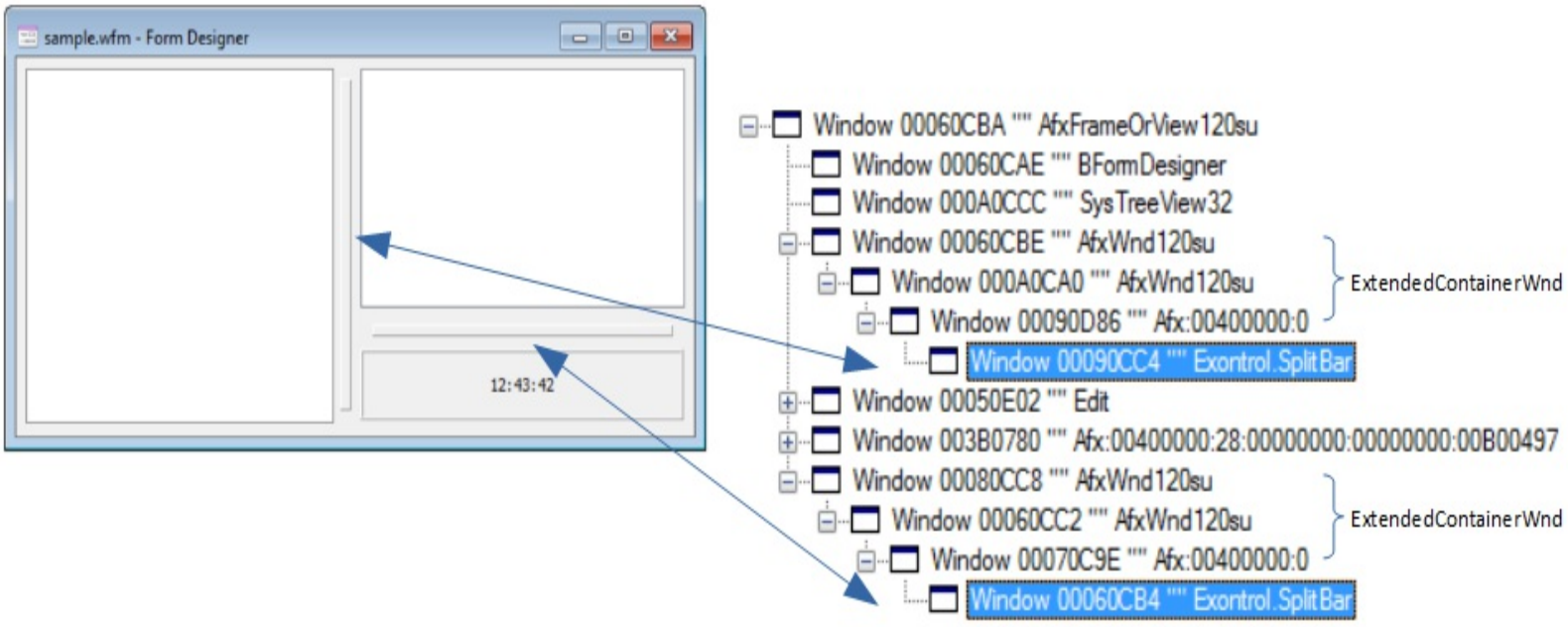
property SplitBar.ExtendedContainerWnd as String

Specifies the list of window class names of parents added by the extended control.

Type	Description
String	A String expression that defines the list of window class names (can include wild characters such as * or ?), separated by backslash / character as explained bellow.

By default, the ExtendedContainerWnd property is "Afx:*\AfxWnd*\AfxWnd*". The ExtendedContainerWnd property has mostly provided for dBASE Plus support, but it may be useful for other non-standard containers or environments. The ExtendedContainerWnd property contains a list of parent class name to be ignored by the component when it performs layout of the other components on the form. Because dBASE Plus environment places every new ActiveX inside a new container, you can not get access through the form's objects using the IOleContainer::EnumObjects, as it would list the component only. In this case, we have provided the ExtendedContainerWnd property that provides support of eXSlitBar component on dBASE Plus environment.

For instance, if having the following form in design mode:



Internally, the hierarchy windows of a eXSplitBar inside a dBASE Plus form shows as bellow.

So as the parents of eXSplitBar are "Afx:...\AfxWnd...\AfxWnd..."and so the ExtendedContainerWnd property is "Afx:*\AfxWnd*\AfxWnd*".

If future generations of dBASE Plus will change the hierarchy of the windows inside the form, the ExtendedContainerWnd property may need to be changed.

property SplitBar.ExtendedHeight as String

Specifies a list of property names separated by comma character, that indicates the Height property of the extended control. The Height property of an extended control gets or sets the control's height.

Type	Description
String	A String expression that specifies a list of property names separated by comma character, that indicates the Height property of the extended control.

By default, the ExtendedHeight property is "Height", which specifies that the split bar is using the "Height" property of the extended control to specify the control's height. Extended controls may be implemented by OLE control containers to provide a wrapper for contained controls.

The /COM version may use the following properties:

- ExtendedLeft property specifies a list of property names separated by comma character, that indicates the Left property of the extended control. The Left property of an extended control gets or sets the distance, between the left edge of the object and the left edge of its container
- ExtendedTop property specifies a list of property names separated by comma character, that indicates the Top property of the extended control. The Top property of an extended control gets or sets the distance, between the top edge of the object and the top edge of its container
- ExtendedWidth property specifies a list of property names separated by comma character, that indicates the Width property of the extended control. The Width property of an extended control gets or sets the control's width
- ExtendedHeight property specifies a list of property names separated by comma character, that indicates the Height property of the extended control. The Height property of an extended control gets or sets the control's height.

to determine/change the location/size of the component on the form/dialog/window.

The /NET version uses the following properties:

- Left, Top, Width and Height to determine the location and size of the component on the form
- SetBounds method to change the location and size of the component on the form

This property is not available for the /NET version.

property SplitBar.ExtendedLeft as String

Specifies a list of property names separated by comma character, that indicates the Left property of the extended control. The Left property of an extended control gets or sets the distance, between the left edge of the object and the left edge of its container

Type	Description
String	A String expression that specifies a list of property names separated by comma character, that indicates the Left property of the extended control.

By default, the ExtendedLeft property is "Left", which specifies that the split bar is using the "Left" property of the extended control to get or set the distance, between the left edge of the object and the left edge of its container. Extended controls may be implemented by OLE control containers to provide a wrapper for contained controls.

The /COM version may use the following properties:

- ExtendedLeft property specifies a list of property names separated by comma character, that indicates the Left property of the extended control. The Left property of an extended control gets or sets the distance, between the left edge of the object and the left edge of its container
- ExtendedTop property specifies a list of property names separated by comma character, that indicates the Top property of the extended control. The Top property of an extended control gets or sets the distance, between the top edge of the object and the top edge of its container
- ExtendedWidth property specifies a list of property names separated by comma character, that indicates the Width property of the extended control. The Width property of an extended control gets or sets the control's width
- ExtendedHeight property specifies a list of property names separated by comma character, that indicates the Height property of the extended control. The Height property of an extended control gets or sets the control's height.

to determine/change the location/size of the component on the form/dialog/window.

The /NET version uses the following properties:

- Left, Top, Width and Height to determine the location and size of the component on the form
- SetBounds method to change the location and size of the component on the form

This property is not available for the /NET version.

property SplitBar.ExtendedName as String

Specifies a list of property names separated by comma character, that indicates the Name property of the extended control. The Name property of an extended control specifies the name of the object within the container.

Type	Description
String	A String expression that specifies a list of property names separated by comma character, that indicates the Name property of the extended control. For instance: "Name,Caption"

By default, the ExtendedName property is "Name". The Name property of an extended control specifies the name of the object within the container. The ExtendedName property Specifies a list of property names separated by comma character, that indicates the Name property of the extended control. The Name property of an extended control specifies the name of the object within the container. Shortly, the ExtendedName property defines the name of the property that can identify the name of the component as known by the container (/COM only).

The ExtendedName property is used by /COM version by the following properties:

- ObjectsLT property indicates the object(s) to be updated in the left/top part of the split bar
- AddObjectLT method adds a new object to be updated in the left/top part of the split bar
- ObjectsLT property indicates the object(s) to be updated in the right/bottom part of the split bar
- AddObjectRB method adds a new object to be updated in the right/bottom part of the split bar

on following programming languages:

- C# for /NET Assembly
- Microsoft Office (Access, Excel, Word)
- Visual Basic 6
- Visual Basic for /NET Assembly
- Visual FoxPro

The ExtendedName property is not available on the /NET version.

property SplitBar.ExtendedObject as String

Specifies a list of property names separated by comma character, that indicates the Object property of the extended control. The Object property of an extended control returns the original /hosted object.

Type	Description
String	A String expression that specifies a list of property names separated by comma character, that indicates the Object property of the extended control. For instance, "Object,GetOcx,DefaultInterface,nativeObject"

By default, the ExtendedObject property is "Object". The Object property of an extended control returns the original /hosted object.

The ExtendedObject property may be used by /COM version by the following properties:

- ObjectsLT property indicates the object(s) to be updated in the left/top part of the split bar
- AddObjectLT method adds a new object to be updated in the left/top part of the split bar
- ObjectsLT property indicates the object(s) to be updated in the right/bottom part of the split bar
- AddObjectRB method adds a new object to be updated in the right/bottom part of the split bar

on following programming languages:

- C# for /NET Assembly
- Microsoft Office (Access, Excel, Word)
- Visual Basic 6
- Visual Basic for /NET Assembly
- Visual FoxPro

The ExtendedObject property is not available on the /NET version.

property SplitBar.ExtendedTop as String

Specifies a list of property names separated by comma character, that indicates the Top property of the extended control. The Top property of an extended control gets or sets the distance, between the top edge of the object and the top edge of its container.

Type	Description
String	A String expression that specifies a list of property names separated by comma character, that indicates the Top property of the extended control.

By default, the ExtendedTop property is "Top", which specifies that the split bar is using the "Top" property of the extended control to get or set the distance, between the top edge of the object and the top edge of its container. Extended controls may be implemented by OLE control containers to provide a wrapper for contained controls.

The /COM version may use the following properties:

- ExtendedLeft property specifies a list of property names separated by comma character, that indicates the Left property of the extended control. The Left property of an extended control gets or sets the distance, between the left edge of the object and the left edge of its container
- ExtendedTop property specifies a list of property names separated by comma character, that indicates the Top property of the extended control. The Top property of an extended control gets or sets the distance, between the top edge of the object and the top edge of its container
- ExtendedWidth property specifies a list of property names separated by comma character, that indicates the Width property of the extended control. The Width property of an extended control gets or sets the control's width
- ExtendedHeight property specifies a list of property names separated by comma character, that indicates the Height property of the extended control. The Height property of an extended control gets or sets the control's height.

to determine/change the location/size of the component on the form/dialog/window.

The /NET version uses the following properties:

- Left, Top, Width and Height to determine the location and size of the component on the form
- SetBounds method to change the location and size of the component on the form

This property is not available for the /NET version.

property SplitBar.ExtendedVisible as String

Specifies a list of property names separated by comma character, that indicates the Visible property of the extended control. The Visible property of an extended control shows or hides object.

Type	Description
String	A String expression that specifies a list of property names separated by comma character, that indicates the Visible property of the extended control.

By default, the ExtendedVisible property is "Visible", which specifies that the split bar is using the "Visible" property of the extended control to show or hide a component on the container. The Visible property of an extended control shows or hides object. Extended controls may be implemented by OLE control containers to provide a wrapper for contained controls. The HideOnLimit property gets or sets a value that indicates whether the splitting objects are hidden when the split bar is closed to its limit. The LimitLT property specifies the expression that determines the limit to drag the splitter to left/top side of its container. The LimitRB property specifies the expression that determines the limit to drag the splitter to right/bottom side of its container.

The /COM version may use one of the following to show / hide the object:

- using the Visible property of the extended control, as indicated by the ExtendedVisible property
- using the ShowWindow API, if the handle of the window can be detected using the IOleWindow::GetWindow, from obj parameter (VC++ environment)
- using the ShowWindow API, if the obj refers to a handle of the window (Delphi enviroment)

The /NET version shows or hides the objects:

- using the Visible property of the Control object (System.Windows.Forms)

The ExtendedVisible property is not available for /NET version.

property SplitBar.ExtendedWidth as String

Specifies a list of property names separated by comma character, that indicates the Width property of the extended control. The Width property of an extended control gets or sets the control's width.

Type	Description
String	A String expression that specifies a list of property names separated by comma character, that indicates the Width property of the extended control.

By default, the ExtendedWidth property is "Width", which specifies that the split bar is using the "Width" property of the extended control to get or set component's width. Extended controls may be implemented by OLE control containers to provide a wrapper for contained controls.

The /COM version may use the following properties:

- ExtendedLeft property specifies a list of property names separated by comma character, that indicates the Left property of the extended control. The Left property of an extended control gets or sets the distance, between the left edge of the object and the left edge of its container
- ExtendedTop property specifies a list of property names separated by comma character, that indicates the Top property of the extended control. The Top property of an extended control gets or sets the distance, between the top edge of the object and the top edge of its container
- ExtendedWidth property specifies a list of property names separated by comma character, that indicates the Width property of the extended control. The Width property of an extended control gets or sets the control's width
- ExtendedHeight property specifies a list of property names separated by comma character, that indicates the Height property of the extended control. The Height property of an extended control gets or sets the control's height.

to determine/change the location/size of the component on the form/dialog/window.

The /NET version uses the following properties:

- Left, Top, Width and Height to determine the location and size of the component on the form
- SetBounds method to change the location and size of the component on the form

This property is not available for the /NET version.

property SplitBar.Font as IFontDisp

Retrieves or sets the control's font.

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

Use the Font property to change the control's font . Use the Refresh method to refresh the control. Use the BeginUpdate and EndUpdate method to maintain performance while multiple changes are performed..

property SplitBar.ForeColor as Color

Specifies the control's foreground color.

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

The ForeColor property specifies the control's foreground color. The BackColor property specifies the control's background color. The SplitBarBackColor property defines the slit bar's visual appearance/background color. The SplitHotBackColor property defines the slit bar's visual appearance/background color, while cursor is hovering the split bar. The BorderWidth property sets or retrieves a value that indicates the border width of the control. The BorderHeight property sets or retrieves a value that indicates the border height of the control.

property SplitBar.HideOnLimit as Boolean

Gets or sets a value that indicates whether the splitting objects are hidden when the split bar is closed to its limit.

Type	Description
Boolean	A Boolean expression that specifies whether the splitting objects are hidden when the split bar is closed to its limit.

By default, the HideOnLimit property is True, which indicates that the splitting objects are hidden when split bar's position is Min or Max. The Min property indicates leftmost/topmost position the split bar can be moved. The Max property indicates rightmost/bottommost position the split bar can be moved. The LimitLT property specifies the expression that determines the limit to drag the splitter to left/top side of its container. The LimitRB property specifies the expression that determines the limit to drag the splitter to right/bottom side of its container. The control fires the Show event when an object requires to be shown or hidden.

The /COM version may use one of the following to show / hide the object:

- using the Visible property of the extended control. The ExtendedVisible property specifies a list of property names separated by comma character, that indicates the Visible property of the extended control. The Visible property of an extended control shows or hides object. By default, the ExtendedVisible property is "Visible".
- using the ShowWindow API, if the handle of the window can be detected using the IOleWindow::GetWindow, from obj parameter (VC++ environment)
- using the ShowWindow API, if the obj refers to a handle of the window (Delphi enviroment)

The /NET version shows or hides the objects:

- using the Visible property of the Control object (System.Windows.Forms)

property SplitBar.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 you can load pictures into the control:

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

property SplitBar.hWnd as Long

Retrieves the control's window handle.

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

The Microsoft Windows operating environment identifies each form 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. Because the value of this property can change while a program is running, you cannot rely on its value (e.g., when stored in a variable).

method SplitBar.Images (Handle as Variant)

Sets at runtime the control's image list. The Handle should be a handle to an Images List Control.

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

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

The control provides an image list window, that's displayed at design time. The ImageSize property defines the size (width/height) of the icons within the control's Images collection. Use the ShowImageList property to hide the image list window, at design time. At design time, the user can add new icons to the control's Images collection, by dragging icon files, exe files, etc, to the images list window. At runtime, the user can use the Images and ReplaceIcon method to change the Images collection. The Images collection is 1 based.

property SplitBar.ImageSize as Long

Retrieves or sets the size of icons the control displays..

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

By default, the ImageSize property is 16 (pixels). The ImageSize property specifies the size of icons being loaded using the Images method. The control's Images collection is cleared if the ImageSize property is changed, so it is recommended to set the ImageSize property before calling the Images method. The ImageSize property defines the size (width/height) of the icons within the control's Images collection. For instance, if the ICO file to load includes different types the one closest with the size specified by ImageSize property is loaded by Images method. The ImageSize property does NOT change the height for the control's font.

property SplitBar.LimitLT as String

Specifies the expression that determines the limit to drag the splitter to left/top side of its container.

Type	Description
String	A string expression that defines the limit moving the split bar to the left/top side.

By default, the LimitLT property is "8D", which indicates 8 dots (8 pixels for DPI settings of 100%, 12 pixels for DPI settings of 150%, , 16 pixels for DPI settings of 200%, and so on). The LimitLT property specifies the expression that determines the limit to drag the splitter to left/top side of its container. The LimitRB property specifies the expression that determines the limit to drag the splitter to right/bottom side of its container. The Min property indicates leftmost/topmost position the split bar can be moved. The Max property indicates rightmost/bottommost position the split bar can be moved.

Any of the following properties can be used to anchor controls/components/objects to the left/top/right/bottom sides of the split bar:

- ObjectsLT property indicates the object(s) to be updated in the left/top part of the split bar
- AddObjectLT method adds a new object to be updated in the left/top part of the split bar
- ObjectsLT property indicates the object(s) to be updated in the right/bottom part of the split bar
- AddObjectRB method adds a new object to be updated in the right/bottom part of the split bar

For instance:

- "0", specifies that no limit is applied
- "8", indicates that the limit is 8 pixels
- "8D", 8 pixels for DPI settings of 100%, 12 pixels for DPI settings of 150%, , 16 pixels for DPI settings of 200%, and so on
- "25%", limits the split bar to a quarter from the full-distance.
- "25% + 8D"

The LimitLT property supports the following operators:

- value**D**, specifies that the value indicates dots instead of pixels. For instance: 10D specifies 10 pixels for DPI settings of 100%, 15 pixels for DPI settings of 150%, , 20 pixels for DPI settings of 200%, and so on
- value**%**, indicates the percent of size to be used. For instance, 50% indicates that half

of the full-range. The % (percent) is applied to the container, if no object is anchored to the split bar, else it is applied to the object with the minimum size.

- **+**, adds two operands. For instance 10 + 25%, indicates 10 pixels plus a quarter from the full-range
- **-**, subtracts two operands
- */*, divides two numbers
- *, multiples two numbers

property SplitBar.LimitRB as String

Specifies the expression that determines the limit to drag the splitter to right/bottom side of its container.

Type	Description
String	A string expression that defines the limit moving the split bar to the left/top side.

By default, the LimitRB property is "8D", which indicates 8 dots (8 pixels for DPI settings of 100%, 12 pixels for DPI settings of 150%, , 16 pixels for DPI settings of 200%, and so on). The LimitRB property specifies the expression that determines the limit to drag the splitter to right/bottom side of its container. The LimitLT property specifies the expression that determines the limit to drag the splitter to left/top side of its container. The Min property indicates leftmost/topmost position the split bar can be moved. The Max property indicates rightmost/bottommost position the split bar can be moved.

Any of the following properties can be used to anchor controls/components/objects to the left/top/right/bottom sides of the split bar:

- ObjectsLT property indicates the object(s) to be updated in the left/top part of the split bar
- AddObjectLT method adds a new object to be updated in the left/top part of the split bar
- ObjectsLT property indicates the object(s) to be updated in the right/bottom part of the split bar
- AddObjectRB method adds a new object to be updated in the right/bottom part of the split bar

For instance:

- "0", specifies that no limit is applied
- "8", indicates that the limit is 8 pixels
- "8D", 8 pixels for DPI settings of 100%, 12 pixels for DPI settings of 150%, , 16 pixels for DPI settings of 200%, and so on
- "25%", limits the split bar to a quarter from the full-distance.
- "25% + 8D"

The LimitRB property supports the following operators:

- value**D**, specifies that the value indicates dots instead of pixels. For instance: 10D specifies 10 pixels for DPI settings of 100%, 15 pixels for DPI settings of 150%, , 20 pixels for DPI settings of 200%, and so on
- value**%**, indicates the percent of size to be used. For instance, 50% indicates that half

of the full-range. The % (percent) is applied to the container, if no object is anchored to the split bar, else it is applied to the object with the minimum size.

- **+**, adds two operands. For instance 10 + 25%, indicates 10 pixels plus a quarter from the full-range
- **-**, subtracts two operands
- */*, divides two numbers
- *, multiples two numbers

property SplitBar.Max as Long

Indicates rightmost/bottommost position the split bar can be moved.

Type	Description
Long	A Long expression that specifies the rightmost/bottommost position the split bar can be moved

The MoveTo method moves programmatically the split bar to specified position. For instance, MoveTo(Max) moves the split bar and its associated objects to the right/bottom-most position. The Max property indicates rightmost/bottommost position the split bar can be moved. The Min property indicates leftmost/topmost position the split bar can be moved. The LimitLT property specifies the expression that determines the limit to drag the splitter to left/top side of its container. The LimitRB property specifies the expression that determines the limit to drag the splitter to right/bottom side of its container. The MoveOnDrop property specifies whether the objects to the left/top and right/bottom of the split bar are moved while dragging or just when the user drops the split bar. The control fires DragStart event when the user clicks the split bar (start dragging the split bar). The Drag event is fired contiguously while the split bar is dragging. The DragEnd event notifies your application that the user releases the split bar (ends dragging the split bar). The Enabled property specifies whether the control is enabled or disabled. The HideOnLimit property gets or sets a value that indicates whether the splitting objects are hidden when the split bar is closed to its limit.

property SplitBar.Min as Long

Indicates leftmost/topmost position the split bar can be moved.

Type	Description
Long	A Long expression that specifies the leftmost/topmost position the split bar can be moved

The MoveTo method moves programmatically the split bar to specified position. For instance, MoveTo(Min) moves the split bar and its associated objects to the left/top-most position. The Min property indicates leftmost/topmost position the split bar can be moved. The Max property indicates rightmost/bottommost position the split bar can be moved. The LimitLT property specifies the expression that determines the limit to drag the splitter to left/top side of its container. The LimitRB property specifies the expression that determines the limit to drag the splitter to right/bottom side of its container. The MoveOnDrop property specifies whether the objects to the left/top and right/bottom of the split bar are moved while dragging or just when the user drops the split bar. The control fires DragStart event when the user clicks the split bar (start dragging the split bar). The Drag event is fired contiguously while the split bar is dragging. The DragEnd event notifies your application that the user releases the split bar (ends dragging the split bar). The Enabled property specifies whether the control is enabled or disabled. The HideOnLimit property gets or sets a value that indicates whether the splitting objects are hidden when the split bar is closed to its limit.

property SplitBar.Mode as SplitBarModeEnum

Retrieves or sets a value that indicates the split bar's mode.

Type	Description
SplitBarModeEnum	A SplitBarModeEnum expression that determines the split bar's mode.

By default, the Mode property exSplitBarAuto, which indicates that the split bar's mode is determines by its size as:

- if the width of the split bar is greater or equal than its height, the Mode property is exSplitBarVert
- if the width of the split bar is less than its height, the Mode property is exSplitBarHorz

Any of the following properties can be used to anchor controls/components/objects to the left/top/right/bottom sides of the split bar:

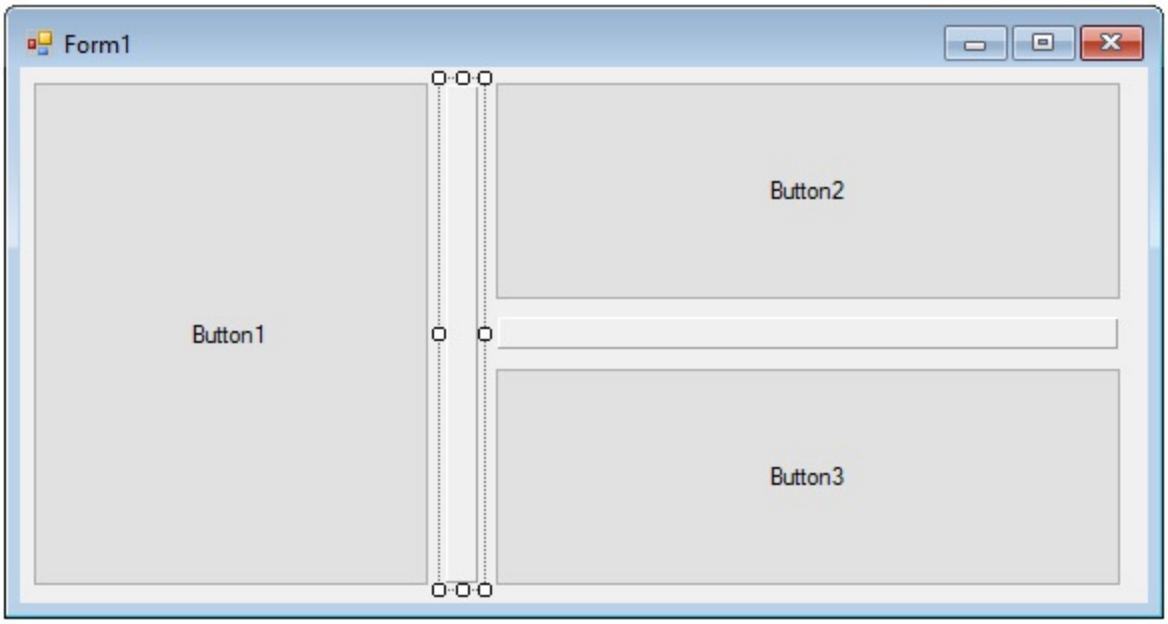
- ObjectsLT property indicates the object(s) to be updated in the left/top part of the split bar
- AddObjectLT method adds a new object to be updated in the left/top part of the split bar
- ObjectsLT property indicates the object(s) to be updated in the right/bottom part of the split bar
- AddObjectRB method adds a new object to be updated in the right/bottom part of the split bar

The LimitRB property specifies the expression that determines the limit to drag the splitter to right/bottom side of its container. The LimitLT property specifies the expression that determines the limit to drag the splitter to left/top side of its container.

Here's the steps you need to follow in order to use the eXSplitBar control:

- Insert the eXSplitBar library/reference to your project
- Place eXSplitBar control to your form/dialog/window
- Specify the objects in the left/top and right/bottom parts of the split bar, using the ObjectsLT / ObjectsRB or AddObjectLT / AddObjectRB available at runtime.

For instance, let's say we have the following layout:

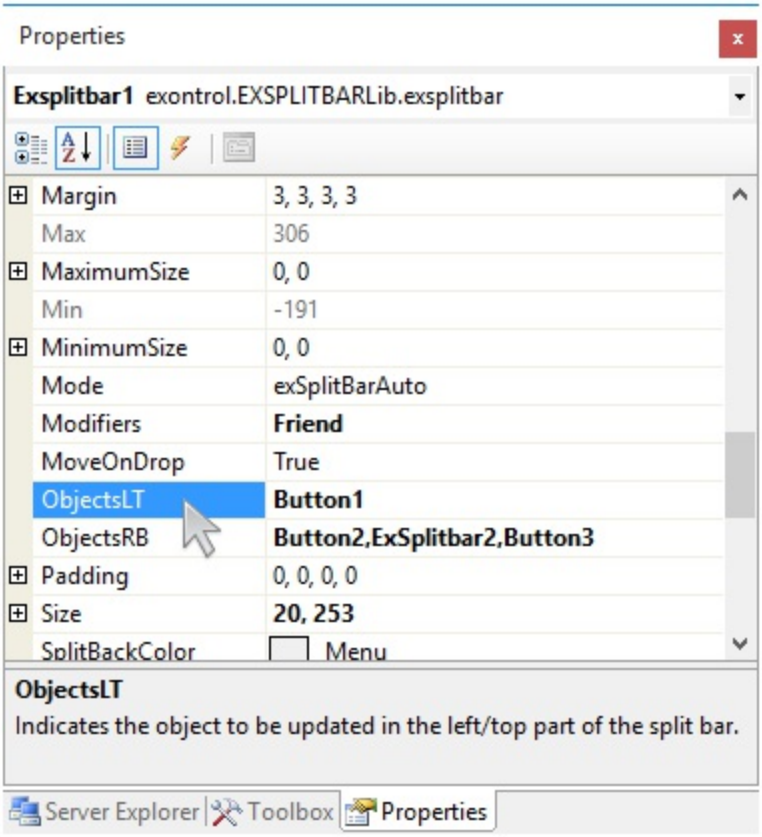


so we have two split bars (horizontal and vertical), and three buttons/commands. One splitter should resizes the left-most button, and the rest, and the vertical split bar show resize the top-most button and the bellow one.

In

- C# for /NET Assembly
- Microsoft Office (Access, Excel, Word)
- Visual Basic 6
- Visual Basic for /NET Assembly
- Visual FoxPro

you can select the split bar in design mode, and specify the objects in the left/top and right/bottom parts of the split bar as in the following screen shot, using the ObjectsLT and ObjectsRB properties:



In

- C++ Builder
- C# for /COM on /NET Framework
- Delphi
- Visual Basic for /COM on /NET Framework
- Visual C++

you need to use the AddObjectLT and AddObjectRB methods as in the following samples.

C++ Builder :

```
SplitBar1->AddObjectLT(TVariant(Button1->Handle));
SplitBar1->AddObjectRB(TVariant(Button2->Handle));
SplitBar1->AddObjectRB(TVariant(SplitBar2->DefaultInterface));
SplitBar1->AddObjectRB(TVariant(Button3->Handle));
```

C# for /COM on /NET Framework :

```
axSplitBar1.AddObjectLT(button1);
axSplitBar1.AddObjectRB(button2);
axSplitBar1.AddObjectRB(axSplitBar2);
axSplitBar1.AddObjectRB(button3);
```

Delphi :

```
with SplitBar1 do
begin
   AddObjectLT(Button1.Handle);
   AddObjectRB(Button2.Handle);
   AddObjectRB(SplitBar2.DefaultInterface);
   AddObjectRB(Button3.Handle);
end
```

Visual Basic for /COM on /NET Framework:

```
With AxSplitBar1
   .AddObjectLT(Button1)
   .AddObjectRB(Button2)
   .AddObjectRB(AxSplitBar2)
   .AddObjectRB(Button3)
End With
```

Visual C++:

```
EXSPLITBARLib::ISplitBarPtr spSplitBar1 = GetDlgItem(IDC_SPLITBAR1)-
>GetControlUnknown();
spSplitBar1->AddObjectLT( (long)::GetDlgItem( m_hWnd, IDC_BUTTON1 ) );
spSplitBar1->AddObjectRB( (long)::GetDlgItem( m_hWnd, IDC_BUTTON2 ) );
spSplitBar1->AddObjectRB( GetDlgItem(IDC_SPLITBAR2)->GetControlUnknown() );
spSplitBar1->AddObjectRB( (long)::GetDlgItem( m_hWnd, IDC_BUTTON3 ) );
```

property SplitBar.MoveOnDrop as Boolean

Gets or sets a value that indicates whether the splitting objects (including the split bar itself) are moved once the user ends dragging the split bar, or contiguously while dragging it.

Type	Description
Boolean	A Boolean expression that indicates whether the splitting objects (including the split bar itself) are moved once the user ends dragging the split bar, or contiguously while dragging it.

By default, the MoveOnDrop property is True, which indicates that the splitting objects (including the split bar itself) are moved once the user drops the split bar. The control fires DragStart event when the user clicks the split bar (start dragging the split bar). The Drag event is fired contiguously while the split bar is dragging. The DragEnd event notifies your application that the user releases the split bar (ends dragging the split bar). The Enabled property specifies whether the control is enabled or disabled. The MoveTo method moves programmatically the split bar to specified position. The MoveOnDrop property specifies whether the objects to the left/top and right/bottom of the split bar are moved while dragging or just when the user drops the split bar.

method SplitBar.MoveTo (Position as Long)

Moves the split bar to the specified position.

Type	Description
Position as Long	A Long expression that specifies the position to move the split bar. The Position property should be a value between Min and Max properties. If 0, the MoveTo method has no effect, just refresh the split bar at the current position.

The MoveTo method moves programmatically the split bar to specified position. For instance, you can programmatically move the split bar to the rightmost / topmost position by calling the MoveTo(Max) method. The Min property indicates leftmost/topmost position the split bar can be moved. The Max property indicates rightmost/bottommost position the split bar can be moved. The LimitLT property specifies the expression that determines the limit to drag the splitter to left/top side of its container. The LimitRB property specifies the expression that determines the limit to drag the splitter to right/bottom side of its container. The MoveOnDrop property specifies whether the objects to the left/top and right/bottom of the split bar are moved while dragging or just when the user drops the split bar. The control fires DragStart event when the user clicks the split bar (start dragging the split bar). The Drag event is fired contiguously while the split bar is dragging. The DragEnd event notifies your application that the user releases the split bar (ends dragging the split bar). The Enabled property specifies whether the control is enabled or disabled.

The /COM version may use the following properties:

- ExtendedLeft property, Specifies a list of property names separated by comma character, that indicates the Left property of the extended control. The Left property of an extended control gets or sets the distance, between the left edge of the object and the left edge of its container.

to determine the location/size of the object in its container.

property SplitBar.ObjectsIN as String

Specifies a list of controls that are child of other controls (prevents changing the left/top part of the control while it is relative to a different parent).

Type	Description
String	A string expression that specifies a list of controls that are child of other controls (prevents changing the left/top part of the control while it is relative to a different parent).

property SplitBar.ObjectsLT as String

Indicates the object to be updated in the left/top part of the split bar.

Type	Description
String	A String expression that specifies the name of the controls on the form to be udated by the split bar when it moves, separated by comma character. For instance, "Command1,Command2"

By default, the ObjectsLT property is "", so no associated components to the split bar. This property is provided to define the controls associated with the left/top side of the split bar at design mode, but it works if using at runtime as well. The Mode property specifies whether the split bar moves objects horizontally or vertically. When the Mode property is set to exSplitBarHorz, the control resizes any controls that lie above or below it, and when the Mode is set to exSplitBarVert, it resizes controls that lie to its left or right. The AddObjectLT method adds at runtime, a new object to be updated in the left/top part of the split bar. The ObjectsRB property defines the objects to be updated on the right/bottom side of the split bar. Setting the ObjectsLT property on "" (empty string), releases any control/object that has been previously anchored to the slit bar, including the objects being added with the AddObjectLT method, or the split bar has nothing attached to its left/top side. The LimitLT property specifies the expression that determines the limit to drag the splitter to left/top side of its container.

By default, if a control/component/object is contained in

- both ObjectsLT / AddObjectLT and ObjectsRB / AddObjectRB, the control/component/object will be moved (not sized), when the split bar moves
- else the control/component/object will be moved and sized accordingly with the side of the split bar it is anchored.

The ExtendedName property Specifies a list of property names separated by comma character, that indicates the Name property of the extended control. The Name property of an extended control specifies the name of the object within the container. The ExtendedName property defines the name of the property that can extracts the name from the container (/COM only).

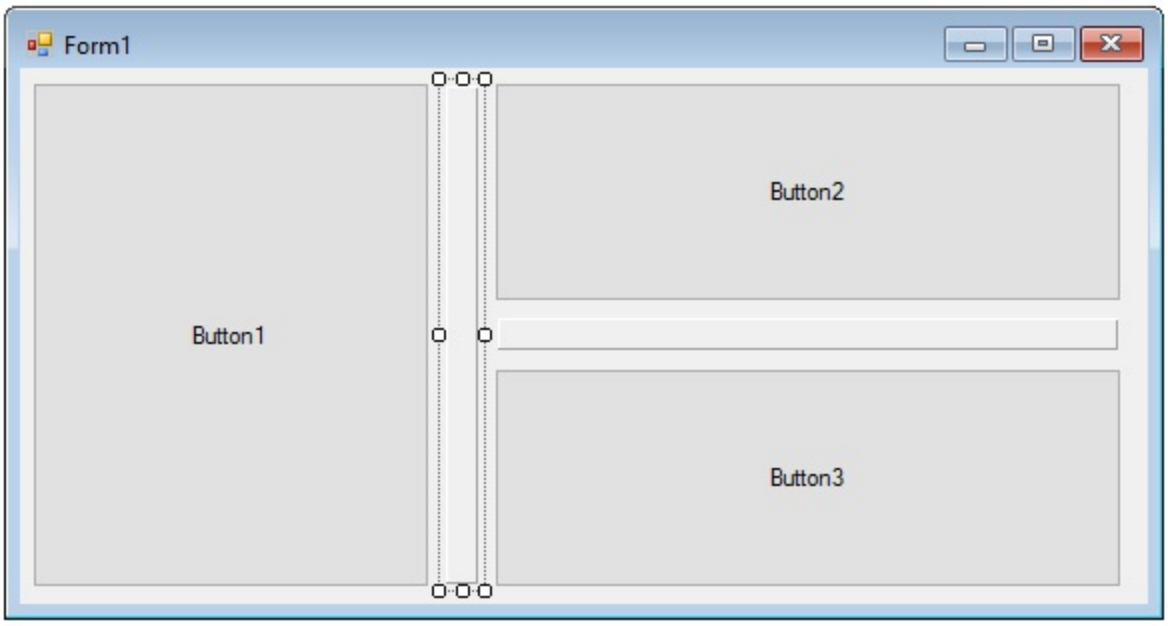
Currently, the ObjectsLT property can be used in any of the following programming environments:

- C# for /NET Assembly
- Microsoft Office (Access, Excel, Word)
- Visual Basic 6
- Visual Basic for /NET Assembly

- Visual FoxPro

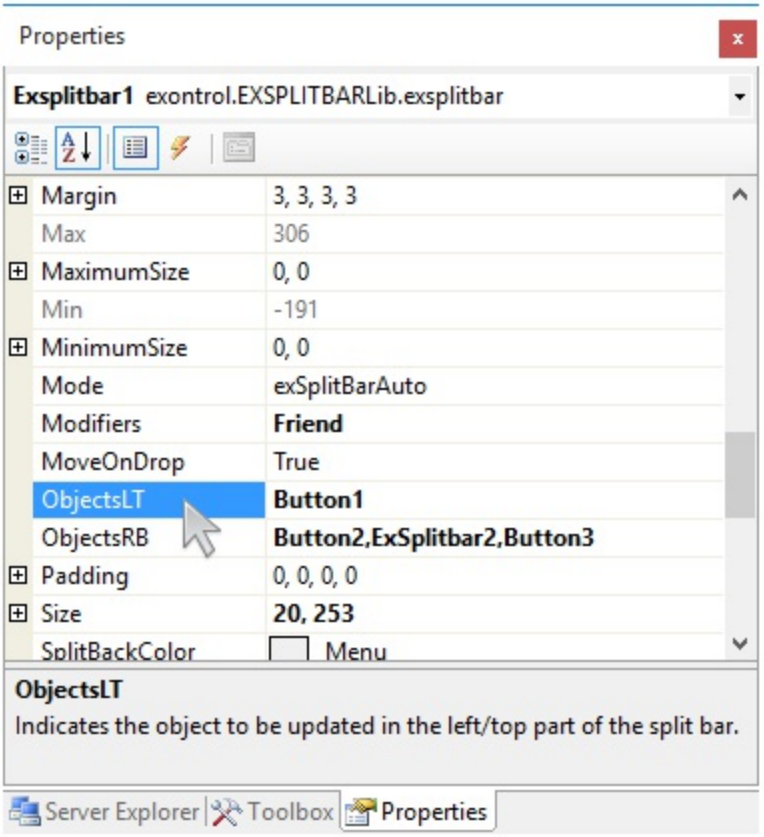
but it could work on other containers as well.

For instance, let's say we have the following layout:



so we have two split bars (horizontal and vertical), and three buttons/commands. One splitter should resizes the left-most button, and the rest, and the vertical split bar show resize the top-most button and the bellow one.

Select the split bar component in design mode and specify the ObjectsLT / ObjectsRB properties as shown bellow:


Properties
Exsplitbar1 exontrol.EXSPLITBARLib.exsplitbar
Margin 3, 3, 3, 3
Max 306
MaximumSize 0, 0
Min -191
MinimumSize 0, 0
Mode exSplitBarAuto
Modifiers Friend
MoveOnDrop True
ObjectsLT Button1
ObjectsRB Button2,ExSplitbar2,Button3
Padding 0, 0, 0, 0
Size 20, 253
SplitBackColor Menu
ObjectsLT
Indicates the object to be updated in the left/top part of the split bar.
Server Explorer
Toolbox
Properties

property SplitBar.ObjectsRB as String

Indicates the object to be updated in the right/bottom part of the split bar.

Type	Description
String	A String expression that specifies the name of the controls on the form to be udated by the split bar when it moves, separated by comma character. For instance, "Command1,Command2"

By default, the ObjectsRB property is "", so no associated components to the split bar. This property is provided to define the controls associated with the right/bottom side of the split bar at design mode, but it works if using at runtime as well. The Mode property specifies whether the split bar moves objects horizontally or vertically. When the Mode property is set to exSplitBarHorz, the control resizes any controls that lie above or below it, and when the Mode is set to exSplitBarVert, it resizes controls that lie to its left or right. The AddObjectRB method adds at runtime, a new object to be updated in the right/bottom part of the split bar. The ObjectsLT property defines the objects to be updated on the left/top side of the split bar. Setting the ObjectsRB property on "" (empty string), releases any control/object that has been previously anchored to the slit bar, including the objects being added with the AddObjectRB method, or the split bar has nothing attached to its right/bottom side. The LimitRB property specifies the expression that determines the limit to drag the splitter to right/bottom side of its container.

By default, if a control/component/object is contained in

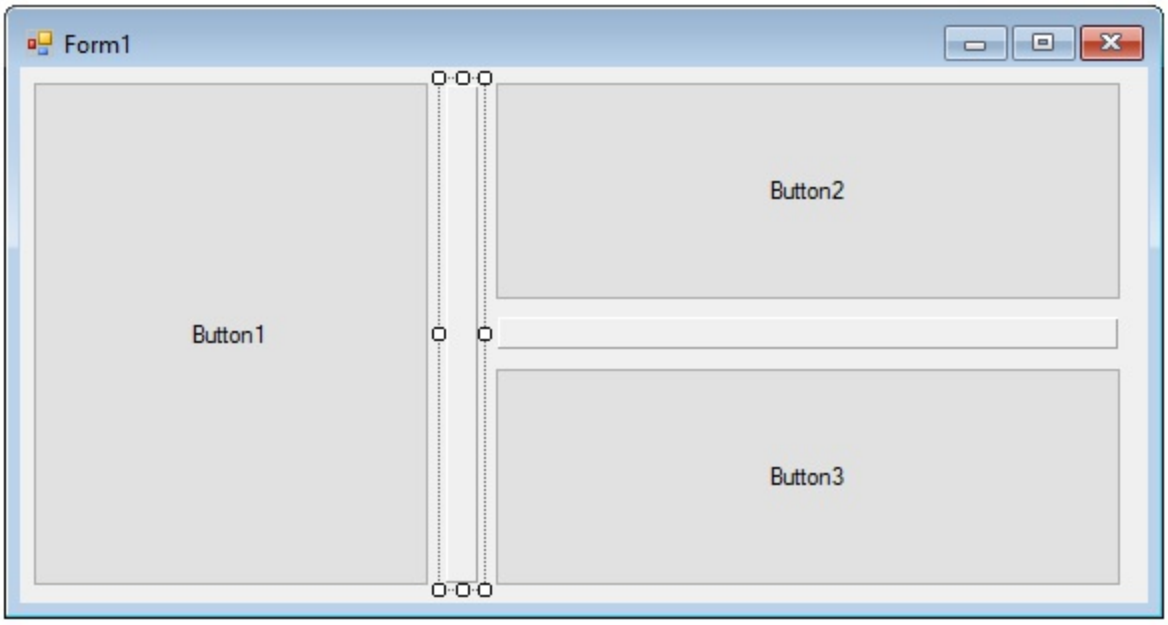
- both ObjectsLT / AddObjectLT and ObjectsRB / AddObjectRB, the control/component/object will be moved (not sized), when the split bar moves
- else the control/component/object will be moved and sized accordingly with the side of the split bar it is anchored.

Currently, the ObjectsRB property can be used in any of the following programming environments:

- C# for /NET Assembly
- Microsoft Office (Access, Excel, Word)
- Visual Basic 6
- Visual Basic for /NET Assembly
- Visual FoxPro

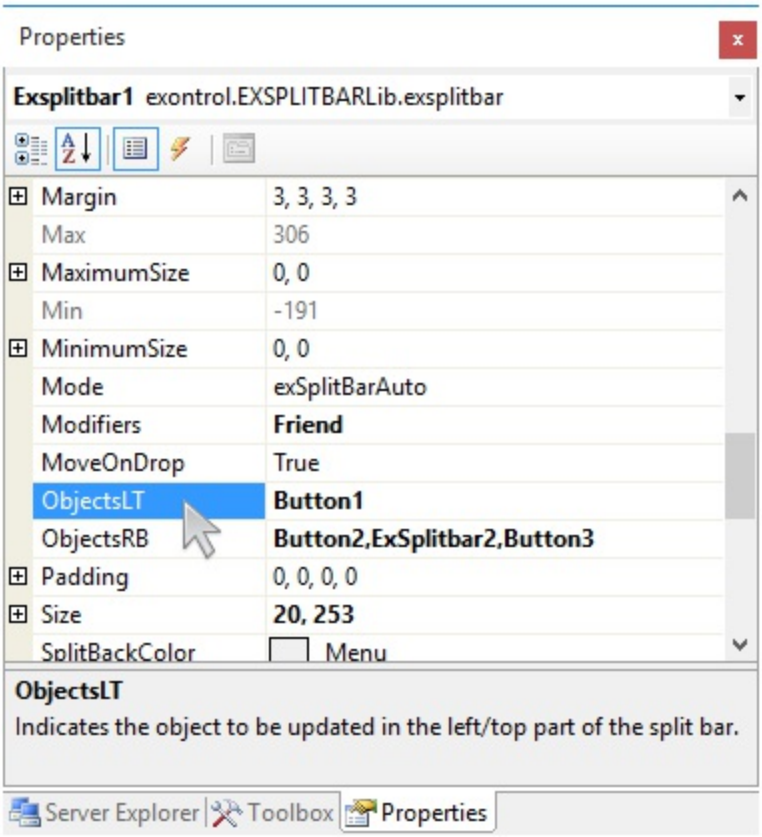
but it could work on other containers as well.

For instance, let's say we have the following layout:



so we have two split bars (horizontal and vertical), and three buttons/commands. One splitter should resizes the left-most button, and the rest, and the vertical split bar show resize the top-most button and the bellow one.

Select the split bar component in design mode and specify the ObjectsLT / ObjectsRB properties as shown bellow:



property SplitBar.Picture as IPictureDisp

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

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

Use the Picture property to load a picture on the control's background. Use the PictureDisplay property to arrange the picture on the control's background.

property SplitBar.PictureDisplay as PictureDisplayEnum

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

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

Use the Picture property to load a picture into the control's background. Use the PictureDisplay property to arrange how the control's picture is displayed on its background. Use the BackColor property to specify the control's background color.

method SplitBar.Refresh ()

Refreshes the control.

Type	Description

The Refresh method refreshes the control. Use BeginUpdate and EndUpdate statement each time when the control requires more changes.

method SplitBar.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 long expression that indicates the icon's handle. By default, the Icon parameter is 0, if it is missing.
Index as Variant	A long expression that indicates the index where icon is inserted. By default, the Index parameter is -1, if it is missing.

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 an image list to the control.

The following sample shows how to add a new icon to control's images list:

```
i = SplitBar1.ReplaceIcon( LoadPicture("d:\icons\help.ico").Handle), in this case the i specifies the index where the icon was added
```

The following sample shows how to replace an icon into control's images list::

```
i = SplitBar1.ReplaceIcon( LoadPicture("d:\icons\help.ico").Handle, 0), in this case the i is zero, because the first icon was replaced.
```

The following sample shows how to remove an icon from control's images list:

```
SplitBar1.ReplaceIcon 0,  i, in this case the i must be the index of the icon that follows to be removed
```

The following sample shows how to clear the control's icons collection:

```
SplitBar1.ReplaceIcon 0,  -1
```

property SplitBar.ShowImageList as Boolean

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

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

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

property SplitBar.SplitBackColor as Color

Specifies the splitter's background color.

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

The SplitBarBackColor property defines the slit bar's visual appearance/background color. The BackColor property specifies the control's background color. The SplitHotBackColor property defines the slit bar's visual appearance/background color, while cursor is hovering the split bar. The BorderWidth property sets or retrieves a value that indicates the border width of the control. The BorderHeight property sets or retrieves a value that indicates the border height of the control.

property SplitBar.SplitHotBackColor as Color

Specifies the splitter's background color, while the cursor is hovering it.

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

The SplitHotBackColor property defines the slit bar's visual appearance/background color, while cursor is hovering the split bar. The SplitBarBackColor property defines the slit bar's visual appearance/background color. The BackColor property specifies the control's background color. The BorderWidth property sets or retrieves a value that indicates the border width of the control. The BorderHeight property sets or retrieves a value that indicates the border height of the control.

property SplitBar.Template as String

Specifies the control's template.

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

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

The Exontrol's eXHelper tool helps you to find easy and quickly the answers and the source code for your questions regarding the usage of our UI components.

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

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

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

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

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

- **Dim** list of variables *Declares the variables. Multiple variables are separated by*

commas. (Sample: Dim h, h1, h2)

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

The x-script may uses constant expressions as follow:

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

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

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

property SplitBar.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 SplitBar.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 SplitBar.ToolTipDelay as Long

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

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

By default, the ToolTipDelay property is 500ms, which indicates that the split bar's tooltip is shown after a half a second. If the ToolTipDelay or ToolTipPopDelay property is 0, the control displays no tooltips. The ToolTipText property specifies the HTML caption to be shown when the cursor is hovering an enabled slit bar. The ToolTipTitle property specifies the title of the control's tooltip. Use the ToolTipPopDelay property specifies the period in ms of time the ToolTip remains visible if the mouse pointer is stationary within a control. Use the ToolTipWidth property to specify the width of the tooltip window. Use the Background(exToolTipAppearance) property indicates the visual appearance of the borders of the tooltips. Use the Background(exToolTipBackColor) property indicates the tooltip's background color. Use the Background(exToolTipForeColor) property indicates the tooltip's foreground color. Use the ToolTipFont property or <font> HTML element to assign a new font for tooltips.

property SplitBar.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 or <font> HTML element to assign a new font for tooltips. If the ToolTipDelay or ToolTipPopDelay property is 0, the control displays no tooltips. The ToolTipText property specifies the HTML caption to be shown when the cursor is hovering an enabled slit bar. The ToolTipTitle property specifies the title of the control's tooltip. Use the ToolTipPopDelay property specifies the period in ms of time the ToolTip remains visible if the mouse pointer is stationary within a control. Use the ToolTipWidth property to specify the width of the tooltip window. Use the Background(exToolTipAppearance) property indicates the visual appearance of the borders of the tooltips. Use the Background(exToolTipBackColor) property indicates the tooltip's background color. Use the Background(exToolTipForeColor) property indicates the tooltip's foreground color.

property SplitBar.ToolTipPopDelay as Long

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

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

By default, the ToolTipPopDelay property is 5000ms, which indicates that the split bar's tooltip is shown for 5 seconds. If the ToolTipDelay or ToolTipPopDelay property is 0, the control displays no tooltips. Use the ToolTipPopDelay property specifies the period in ms of time the ToolTip remains visible if the mouse pointer is stationary within a control. The ToolTipText property specifies the HTML caption to be shown when the cursor is hovering an enabled slit bar. The ToolTipTitle property specifies the title of the control's tooltip. Use the ToolTipWidth property to specify the width of the tooltip window. Use the Background(exToolTipAppearance) property indicates the visual appearance of the borders of the tooltips. Use the Background(exToolTipBackColor) property indicates the tooltip's background color. Use the Background(exToolTipForeColor) property indicates the tooltip's foreground color. Use the ToolTipFont property or HTML element to assign a new font for tooltips.

property SplitBar.ToolTipText as String

Specifies the control's tooltip text.

Type	Description
String	A string expression that indicates the split bar's tooltip.

By default, the ToolTipText property is "". The ToolTipText property specifies the HTML caption to be shown when the cursor is hovering an enabled slit bar. The ToolTipTitle property specifies the title of the control's tooltip. If the ToolTipDelay or ToolTipPopDelay property is 0, the control displays no tooltips. Use the ToolTipPopDelay property specifies the period in ms of time the ToolTip remains visible if the mouse pointer is stationary within a control. Use the ToolTipWidth property to specify the width of the tooltip window. Use the Background(exToolTipAppearance) property indicates the visual appearance of the borders of the tooltips. Use the Background(exToolTipBackColor) property indicates the tooltip's background color. Use the Background(exToolTipForeColor) property indicates the tooltip's foreground color. Use the ToolTipFont property or HTML element to assign a new font for tooltips.

The tooltip supports the following HTML tags:

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

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

- **<dotline rrggbb> ... </dotline>** or <dotline=rrggbb> ... </dotline> draws a dot-line on the bottom side of the current text-line, of specified RGB color. The <dotline> ... </dotline> draws a black dot-line on the bottom side of the current text-line. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<upline> ... </upline>** draws the line on the top side of the current text-line (requires <solidline> or <dotline>).
- **<r>** right aligns the text
- **<c>** centers the text
- **
** forces a line-break
- **<img>number[:width]</img>** inserts an icon inside the text. The number indicates the index of the icon being inserted. Use the Images method to assign a list of icons to your chart. The last 7 bits in the high significant byte of the number expression indicates the identifier of the skin being used to paint the object. Use the Add method to add new skins to the control. If you need to remove the skin appearance from a part of the control you need to reset the last 7 bits in the high significant byte of the color being applied to the part. The width is optional and indicates the width of the icon being inserted. Using the width option you can overwrite multiple icons getting a nice effect. By default, if the width field is missing, the width is 18 pixels.
- **<img>key[:width]</img>** inserts a custom size picture into the text being previously loaded using the HTMLPicture property. The Key parameter indicates the key of the picture being displayed. The Width parameter indicates a custom size, if you require to stretch the picture, else the original size of the picture is used.
- **&** glyph characters as **&** (&), **<** (<), **>** (>), **&qout;** (") and **&#number;** (the character with specified code), For instance, the € displays the EUR character. The **&** ampersand is only recognized as markup when it is followed by a known letter or a #character and a digit. For instance if you want to display bold in HTML caption you can use bold
- **<off offset> ... </off>** defines the vertical offset to display the text/element. The offset parameter defines the offset to display the element. This tag is inheritable, so the offset is keep while the associated </off> tag is found. You can use the <off offset> HTML tag in combination with the to define a smaller or a larger font to be displayed. For instance: "Text with <**off** 6>subscript" displays the text such as: Text with $_{subscript}$ The "Text with <**off** -6>superscript" displays the text such as: Text with ^{subscript}
- **<gra rrggbb;mode;blend> ... </gra>** defines a gradient text. The text color or <fgcolor> defines the starting gradient color, while the rr/gg/bb represents the red/green/blue values of the ending color, 808080 if missing as gray. The mode is a value between 0 and 4, 1 if missing, and blend could be 0 or 1, 0 if missing. The HTML tag can be used to define the height of the font. Any of the rrggbb, mode or

blend field may not be specified. The <gra> with no fields, shows a vertical gradient color from the current text color to gray (808080). For instance the "<**gra** FFFFFF;1;1>gradient-center</**gra**>" generates the following picture:

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

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

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

property SplitBar.ToolTipTitle as String

Specifies the title of the control's tooltip.

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

By default, the ToolTipTitle property is "". The ToolTipTitle property specifies the title of the control's tooltip. The ToolTipText property specifies the HTML caption to be shown when the cursor is hovering an enabled slit bar. If the ToolTipDelay or ToolTipPopDelay property is 0, the control displays no tooltips. Use the ToolTipPopDelay property specifies the period in ms of time the ToolTip remains visible if the mouse pointer is stationary within a control. Use the ToolTipWidth property to specify the width of the tooltip window. Use the Background(exToolTipAppearance) property indicates the visual appearance of the borders of the tooltips. Use the Background(exToolTipBackColor) property indicates the tooltip's background color. Use the Background(exToolTipForeColor) property indicates the tooltip's foreground color. Use the ToolTipFont property or <font> HTML element to assign a new font for tooltips.

property SplitBar.ToolTipWidth as Long

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

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

By default, the ToolTipWidth property is 196 pixels, which indicates that the width of the split bar's tooltip is not greater than 196 pixels. Use the ToolTipWidth property to specify the width of the tooltip window. If the ToolTipDelay or ToolTipPopDelay property is 0, the control displays no tooltips. The ToolTipText property specifies the HTML caption to be shown when the cursor is hovering an enabled slit bar. The ToolTipTitle property specifies the title of the control's tooltip. Use the ToolTipPopDelay property specifies the period in ms of time the ToolTip remains visible if the mouse pointer is stationary within a control. Use the Background(exToolTipAppearance) property indicates the visual appearance of the borders of the tooltips. Use the Background(exToolTipBackColor) property indicates the tooltip's background color. Use the Background(exToolTipForeColor) property indicates the tooltip's foreground color. Use the ToolTipFont property or <font> HTML element to assign a new font for tooltips.

property SplitBar.Version as String

Retrieves the control's version.

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

The Version property specifies the control's version.

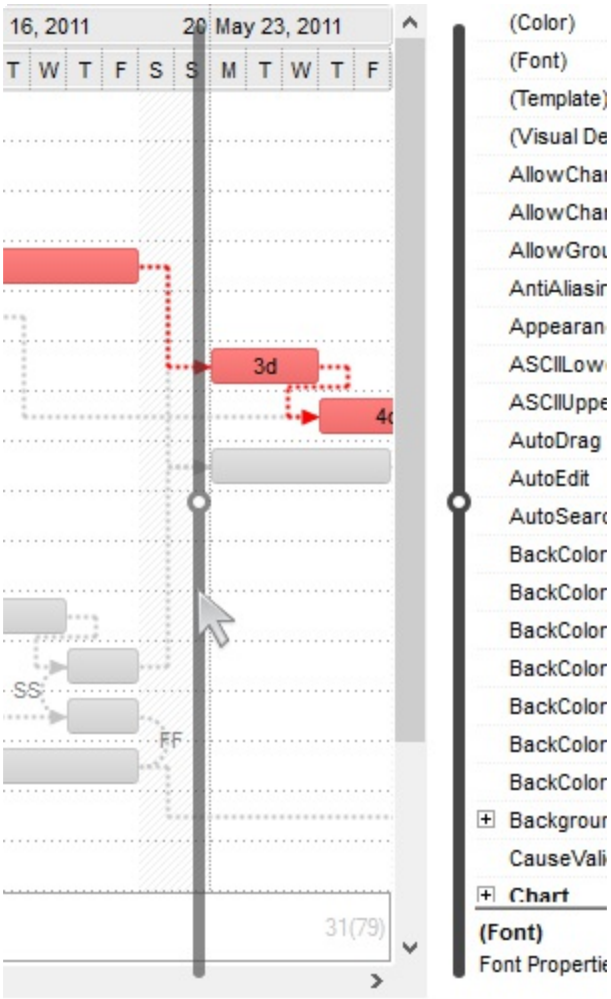
property SplitBar.VisualAppearance as Appearance

Retrieves the control's appearance.

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

Use the Add method to add or replace skins to the control. The skin method, in it's simplest form, uses a single graphic file (*.ebn) assigned to a part of the control. By using a collection of objects laid over the graphic, it is possible to define which sections of the graphic will be used as borders, corners and other possible elements, fixing them to their proper position regardless of the size of the part. The SplitBarBackColor property defines the slit bar's visual appearance/background color. The SplitHotBackColor property defines the slit bar's visual appearance/background color, while cursor is hovering the split bar. The BorderWidth property sets or retrieves a value that indicates the border width of the control. The BorderHeight property sets or retrieves a value that indicates the border height of the control.

The following screen shot shows the split bar with an EBN object:



The following screen shot shows the split bar with a solid color:


16, 2011
20 May 23, 2011
T W T F S S M T W T F
3d
SS
FF
16(39)
(Color)
(Font)
(Template)
(Visual De
AllowCha
AllowCha
AllowGrou
AntiAliasir
Appearan
ASCIILow
ASCIIUppe
AutoDrag
AutoEdit
AutoSear
BackColor
BackColor
BackColor
BackColor
BackColor
BackColor
BackColor
Backgrou
CauseVali
Chart
(Font)
Font Properti

ExSplitBar 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: {9F28FDED-5EBC-4E9A-A596-C3813C966A0C}. The object's program identifier is: "Exontrol.SplitBar". The /COM object module is: "ExSplitBar.dll"

The ExSplitBar component supports the following events:

Name	Description
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.
Drag	Notifies that the user drags the split bar.
DragEnd	Occurs once the user ends dragging the split bar.
DragStart	Occurs once the user starts dragging the split bar.
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.
Show	Occurs when an object requires to be shown or hidden.

event Click ()

Occurs when the user presses and then releases the left mouse button over the control.

Type	Description

The Click event is fired when the user releases the left mouse button over the control. Use a MouseDown or MouseUp event procedure to specify actions that will occur when a mouse button is pressed or released. Unlike the Click and DblClick events, MouseDown and MouseUp events lets you distinguish between the left, right, and middle mouse buttons. You can also write code for mouse-keyboard combinations that use the SHIFT, CTRL, and ALT keyboard modifiers. The control fires the DragStart event when user starts dragging the split bar, Drag event while dragging it, and DragEnd event when dragging the split bar ends.

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(oSplitBar)

RETURN
```

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

Java…

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

VBSc…

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

Visual Data…

```
Procedure OnComClick
    Forward Send OnComClick
End_Procedure
```

Visual Objects

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

X++

```
void onEvent_Click()
{
}
```

XBasic

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

dBASE

```
function nativeObject_Click()
return
```

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

Occurs when the user dblclk the left mouse button over an object.

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

The DblClick event is fired when user double clicks the control. Use a MouseDown or MouseUp event procedure to specify actions that will occur when a mouse button is pressed or released. Unlike the Click and DblClick events, MouseDown and MouseUp events lets you distinguish between the left, right, and middle mouse buttons. You can also write code for mouse-keyboard combinations that use the SHIFT, CTRL, and ALT keyboard modifiers. The control fires the DragStart event when user starts dragging the split bar, Drag event while dragging it, and DragEnd event when dragging the split bar ends.

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

C#

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

VB

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

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

C#

```
private void DblClick(object sender,
AxEXSPLITBARLib._ISplitBarEvents_DblClickEvent e)
{
}
```

C++

```
void OnDblClick(short   Shift,long   X,long   Y)
```

```
{
}
```

C++ Builder

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

Delphi

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

Delphi 8 (.NET only)

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

Powe…

```
begin event DblClick(integer  Shift,long  X,long  Y)

end event DblClick
```

VB.NET

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

VB6

```
Private Sub DblClick(Shift As Integer,X As Single,Y As Single)
End Sub
```

VBA

```
Private Sub DblClick(ByVal Shift As Integer,ByVal X As Long,ByVal Y As Long)
End Sub
```

VFP

```
LPARAMETERS Shift,X,Y

```

Xbas…

```
PROCEDURE OnDblClick(oSplitBar,Shift,X,Y)

RETURN
```

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

Java…

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

VBSc…

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

Visual Data…

```
Procedure OnComDblClick Short   llShift OLE_XPOS_PIXELS   llX OLE_YPOS_PIXELS
llY
    Forward Send OnComDblClick llShift llX llY
End_Procedure
```

Visual Objects

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

X++

```
void onEvent_DblClick(int   _Shift,int   _X,int   _Y)
{
}
```

XBasic

```
function DblClick as v (Shift  as  N,X  as
OLE::Exontrol.SplitBar.1::OLE_XPOS_PIXELS,Y  as
OLE::Exontrol.SplitBar.1::OLE_YPOS_PIXELS)
end function
```

dBASE

```
function nativeObject_DblClick(Shift,X,Y)
return
```

event Drag (Position as Long)

Notifies that the user drags the split bar.

Type	Description
Position as Long	A long expression that specifies position the split bar has been moved by dragging. It indicates the distance in pixels, from the point where the dragging begins.

The Drag event is fired contiguously while the split bar is dragging. The control fires DragStart event when the user clicks the split bar (start dragging the split bar). The DragEnd event notifies your application that the user releases the split bar (ends dragging the split bar). The Enabled property specifies whether the control is enabled or disabled. The MoveTo method moves programmatically the split bar to specified position. The MoveOnDrop property specifies whether the objects to the left/top and right/bottom of the split bar are moved while dragging or just when the user drops the split bar.

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

C#

```
private void Drag(object sender,int   Position)
{
}
```

VB

```
Private Sub Drag(ByVal sender As System.Object,ByVal Position As Integer)
Handles Drag
End Sub
```

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

C#

```
private void Drag(object sender, AxEXSPLITBARLib._ISplitBarEvents_DragEvent e)
{
}
```

C++

```
void OnDrag(long   Position)
{
}
```

C++ Builder

```
void __fastcall Drag(TObject *Sender,long   Position)
{
}
```

Delphi

```
procedure Drag(ASender: TObject; Position : Integer);
begin
end;
```

Delphi 8 (.NET only)

```
procedure Drag(sender: System.Object; e:
AxEXSPLITBARLib._ISplitBarEvents_DragEvent);
begin
end;
```

Powe…

```
begin event Drag(long  Position)


end event Drag
```

VB.NET

```
Private Sub Drag(ByVal sender As System.Object, ByVal e As
AxEXSPLITBARLib._ISplitBarEvents_DragEvent) Handles Drag
End Sub
```

VB6

```
Private Sub Drag(ByVal Position As Long)
End Sub
```

VBA

```
Private Sub Drag(ByVal Position As Long)
End Sub
```

VFP

```
LPARAMETERS Position
```

Xbas…

```
PROCEDURE OnDrag(oSplitBar,Position)

RETURN
```

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

Java…

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

VBSc…

```
<SCRIPT LANGUAGE="VBScript">
Function Drag(Position)
```

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

Visual Data…

```
Procedure OnComDrag Integer   llPosition
    Forward Send OnComDrag llPosition
End_Procedure
```

Visual Objects

```
METHOD OCX_Drag(Position) CLASS MainDialog
RETURN NIL
```

X++

```
void onEvent_Drag(int   _Position)
{
}
```

XBasic

```
function Drag as v (Position  as  N)
end function
```

dBASE

```
function nativeObject_Drag(Position)
return
```

event DragEnd (Position as Long, Cancel as Boolean)

Occurs once the user ends dragging the split bar.

Type	Description
Position as Long	A long expression that determines the position where the drag operation ends. This 0 indicates the initial position, or where the dragging begins.
Cancel as Boolean	A Boolean expression that specifies whether the dragging operation has been canceled or not.

The DragEnd event notifies your application that the user releases the split bar (ends dragging the split bar). The control fires DragStart event when the user clicks the split bar (start dragging the split bar). The Drag event is fired contiguously while the split bar is dragging. The Enabled property specifies whether the control is enabled or disabled. The MoveTo method moves programmatically the split bar to specified position. The MoveOnDrop property specifies whether the objects to the left/top and right/bottom of the split bar are moved while dragging or just when the user drops the split bar.

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

C#
```
private void DragEnd(object sender,int  Position,bool  Cancel)
{
}
```

VB
```
Private Sub DragEnd(ByVal sender As System.Object,ByVal Position As
Integer,ByVal Cancel As Boolean) Handles DragEnd
End Sub
```

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

C#
```
private void DragEnd(object sender,
AxEXSPLITBARLib._ISplitBarEvents_DragEndEvent e)
{
}
```

C++
```
void OnDragEnd(long  Position,BOOL  Cancel)
{
}
```

C++ Builder

```
void __fastcall DragEnd(TObject *Sender,long   Position,VARIANT_BOOL   Cancel)
{
}
```

Delphi

```
procedure DragEnd(ASender: TObject; Position : Integer;Cancel : WordBool);
begin
end;
```

Delphi 8 (.NET only)

```
procedure DragEnd(sender: System.Object; e:
AxEXSPLITBARLib._ISplitBarEvents_DragEndEvent);
begin
end;
```

Powe…

```
begin event DragEnd(long  Position,boolean  Cancel)

end event DragEnd
```

VB.NET

```
Private Sub DragEnd(ByVal sender As System.Object, ByVal e As
AxEXSPLITBARLib._ISplitBarEvents_DragEndEvent) Handles DragEnd
End Sub
```

VB6

```
Private Sub DragEnd(ByVal Position As Long,ByVal Cancel As Boolean)
End Sub
```

VBA

```
Private Sub DragEnd(ByVal Position As Long,ByVal Cancel As Boolean)
End Sub
```

VFP

```
LPARAMETERS Position,Cancel
```

Xbas…

```
PROCEDURE OnDragEnd(oSplitBar,Position,Cancel)

RETURN
```

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

Java…

```
<SCRIPT EVENT="DragEnd(Position,Cancel)" LANGUAGE="JScript">
</SCRIPT>
```

VBSc…

```
<SCRIPT LANGUAGE="VBScript">
Function DragEnd(Position,Cancel)
End Function
</SCRIPT>
```

Visual Data…

```
Procedure OnComDragEnd Integer   llPosition Boolean   llCancel
    Forward Send OnComDragEnd llPosition llCancel
End_Procedure
```

Visual Objects

```
METHOD OCX_DragEnd(Position,Cancel) CLASS MainDialog
RETURN NIL
```

X++

```
void onEvent_DragEnd(int   _Position,boolean   _Cancel)
{
}
```

XBasic

```
function DragEnd as v (Position  as  N,Cancel  as  L)
end function
```

dBASE

```
function nativeObject_DragEnd(Position,Cancel)
return
```

event DragStart (Position as Long, Cancel as Boolean)

Occurs once the user starts dragging the split bar.

Type	Description
Position as Long	A long expression that determines the position where the drag operation begins. This parameter is always 0.
Cancel as Boolean	A Boolean expression that specifies whether the drag operation should continue or cancel. You can change the Cancel parameter during the DragStart event.

The control fires DragStart event when the user clicks the split bar (start dragging the split bar). The Drag event is fired contiguously while the split bar is dragging. The DragEnd event notifies your application that the user releases the split bar (ends dragging the split bar). The Enabled property specifies whether the control is enabled or disabled. The MoveTo method moves programmatically the split bar to specified position. The MoveOnDrop property specifies whether the objects to the left/top and right/bottom of the split bar are moved while dragging or just when the user drops the split bar.

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

C#
```
private void DragStart(object sender,int  Position,ref bool  Cancel)
{
}
```

VB
```
Private Sub DragStart(ByVal sender As System.Object,ByVal Position As
Integer,ByRef Cancel As Boolean) Handles DragStart
End Sub
```

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

C#
```
private void DragStart(object sender,
AxEXSPLITBARLib._ISplitBarEvents_DragStartEvent e)
{
}
```

C++
```
void OnDragStart(long  Position,BOOL FAR*  Cancel)
{
}
```

C++ Builder

```
void __fastcall DragStart(TObject *Sender,long   Position,VARIANT_BOOL *   Cancel)
{
}
```

Delphi

```
procedure DragStart(ASender: TObject; Position : Integer;var Cancel : WordBool);
begin
end;
```

Delphi 8 (.NET only)

```
procedure DragStart(sender: System.Object; e:
AxEXSPLITBARLib._ISplitBarEvents_DragStartEvent);
begin
end;
```

Powe…

```
begin event DragStart(long  Position,boolean  Cancel)

end event DragStart
```

VB.NET

```
Private Sub DragStart(ByVal sender As System.Object, ByVal e As
AxEXSPLITBARLib._ISplitBarEvents_DragStartEvent) Handles DragStart
End Sub
```

VB6

```
Private Sub DragStart(ByVal Position As Long,Cancel As Boolean)
End Sub
```

VBA

```
Private Sub DragStart(ByVal Position As Long,Cancel As Boolean)
End Sub
```

VFP

```
LPARAMETERS Position,Cancel
```

Xbas…

```
PROCEDURE OnDragStart(oSplitBar,Position,Cancel)

RETURN
```

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

Java…

```
<SCRIPT EVENT="DragStart(Position,Cancel)" LANGUAGE="JScript">
</SCRIPT>
```

VBSc…

```
<SCRIPT LANGUAGE="VBScript">
Function DragStart(Position,Cancel)
End Function
</SCRIPT>
```

Visual Data…

```
Procedure OnComDragStart Integer   llPosition Boolean   llCancel
    Forward Send OnComDragStart llPosition llCancel
End_Procedure
```

Visual Objects

```
METHOD OCX_DragStart(Position,Cancel) CLASS MainDialog
RETURN NIL
```

X++

```
void onEvent_DragStart(int   _Position,COMVariant /*bool*/   _Cancel)
{
}
```

XBasic

```
function DragStart as v (Position  as  N,Cancel  as  L)
end function
```

dBASE

```
function nativeObject_DragStart(Position,Cancel)
return
```

event Event (EventID as Long)

Notifies the application once the control fires an event.

Type	Description
EventID as Long	A Long expression that specifies the identifier of the event. Each internal event of the control has an unique identifier. Use the EventParam(-2) to display entire information about fired event (such as name, identifier, and properties). The EventParam(-1) retrieves the number of parameters of fired event

The Event notification occurs ANY time the control fires an event. *This is useful for X++, which does not support event with parameters passed by reference. Also, this could be useful for C++ Builder or Delphi, which does not handle properly the events with parameters of VARIANT type.*

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

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

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 exsplitbar1.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( 0 , 0 , 19 , 191 )
MouseDown/-605( 1 , 0 , 19 , 191 )
DragStart/4( 0 , =false )
MouseMove/-606( 1 , 0 , 20 , 191 )
```

```
Drag/5( 1 )
MouseMove/-606( 1 , 0 , 21 , 191 )
Drag/5( 3 )
MouseMove/-606( 1 , 0 , 22 , 191 )
Drag/5( -6 )
DragEnd/6( -6 , false )
MouseUp/-607( 1 , 0 , 19 , 194 )
MouseMove/-606( 0 , 0 , 24 , 190 )
```

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.

In conclusion, anytime the X++ fires the "invalid parameters." while handling an event, you can use and handle the Event notification and EventParam methods of the control

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

C#
```
private void Event(object sender,int   EventID)
{
}
```

VB
```
Private Sub Event(ByVal sender As System.Object,ByVal EventID As Integer)
Handles Event
End Sub
```

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

C#
```
private void Event(object sender, AxEXSPLITBARLib._ISplitBarEvents_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:
AxEXSPLITBARLib._ISplitBarEvents_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
AxEXSPLITBARLib._ISplitBarEvents_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(oSplitBar,EventID)

RETURN
```

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

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

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

```
</SCRIPT>
```

Visual Data…

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

Visual Objects

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

X++

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

XBasic

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

dBASE

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

event KeyDown (KeyCode as Integer, Shift as Integer)

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

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

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

ShiftDown = (Shift And 1) > 0
CtrlDown = (Shift And 2) > 0
AltDown = (Shift And 4) > 0
In a procedure, you can test for any combination of conditions, as in this example:
If AltDown And CtrlDown Then

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

C#

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

VB

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

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

C#

```
private void KeyDownEvent(object sender,
AxEXSPLITBARLib._ISplitBarEvents_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:
AxEXSPLITBARLib._ISplitBarEvents_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
AxEXSPLITBARLib._ISplitBarEvents_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(oSplitBar,KeyCode,Shift)
```

```
RETURN
```

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

Java…

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

VBSc…

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

Visual Data…

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

Visual Objects

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

X++

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

XBasic

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

dBASE

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

event KeyPress (KeyAscii as Integer)

Occurs when the user presses and releases an ANSI key.

Type	Description
KeyAscii as Integer	An integer that returns a standard numeric ANSI keycode

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

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

C#

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

VB

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

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

C#

```
private void KeyPressEvent(object sender,
AxEXSPLITBARLib._ISplitBarEvents_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:
AxEXSPLITBARLib._ISplitBarEvents_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
AxEXSPLITBARLib._ISplitBarEvents_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(oSplitBar,KeyAscii)


RETURN
```

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

Java…

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

VBSc…

```
<SCRIPT LANGUAGE="VBScript">
Function KeyPress(KeyAscii)
```

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

Visual Data…

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

Visual Objects

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

X++

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

XBasic

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

dBASE

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

event KeyUp (KeyCode as Integer, Shift as Integer)

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

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

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

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

C#

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

VB

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

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

C#

```
private void KeyUpEvent(object sender,
AxEXSPLITBARLib._ISplitBarEvents_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:
AxEXSPLITBARLib._ISplitBarEvents_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
AxEXSPLITBARLib._ISplitBarEvents_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(oSplitBar,KeyCode,Shift)

RETURN
```

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

Java…

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

VBSc…

```
<SCRIPT LANGUAGE="VBScript">
```

```
Function KeyUp(KeyCode,Shift)
End Function
</SCRIPT>
```

Visual Data…

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

Visual Objects

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

X++

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

XBasic

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

dBASE

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

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

Occurs when the user presses a mouse button.

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

Use a MouseDown or MouseUp event procedure to specify actions that will occur when a mouse button is pressed or released. Unlike the Click and DblClick events, MouseDown and MouseUp events lets you distinguish between the left, right, and middle mouse buttons. You can also write code for mouse-keyboard combinations that use the SHIFT, CTRL, and ALT keyboard modifiers. The control fires the DragStart event when user starts dragging the split bar, Drag event while dragging it, and DragEnd event when dragging the split bar ends.

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, AxEXSPLITBARLib._ISplitBarEvents_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:
AxEXSPLITBARLib._ISplitBarEvents_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
AxEXSPLITBARLib._ISplitBarEvents_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(oSplitBar,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.SplitBar.1::OLE_XPOS_PIXELS,Y  as
OLE::Exontrol.SplitBar.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. The control fires the DragStart event when user starts dragging the split bar, Drag event while dragging it, and DragEnd event when dragging the split bar ends.

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, AxEXSPLITBARLib._ISplitBarEvents_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:
AxEXSPLITBARLib._ISplitBarEvents_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
AxEXSPLITBARLib._ISplitBarEvents_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(oSplitBar,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.SplitBar.1::OLE_XPOS_PIXELS,Y  as
OLE::Exontrol.SplitBar.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,
AxEXSPLITBARLib._ISplitBarEvents_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:
AxEXSPLITBARLib._ISplitBarEvents_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
AxEXSPLITBARLib._ISplitBarEvents_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(oSplitBar,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.SplitBar.1::OLE_XPOS_PIXELS,Y  as
OLE::Exontrol.SplitBar.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

The RClick event notifies your application when the user right clicks the control. By default, the user can drag the split bar by left or right click the split bar. In order to prevent dragging the split bar when using the right mouse button, you can change the Cancel parameter of the DragStart event. Use the RClick event to add your context menu. Use the Click event to notify your application that the user clicks the control (using the left mouse button). Use the MouseDown or MouseUp event if you require the cursor position during the RClick event.

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

C#

```
private void RClick(object sender)
{
}
```

VB

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

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

C#

```
private void RClick(object sender, EventArgs e)
{
}
```

C++

```
void OnRClick()
{
}
```

C++ Builder

```
void __fastcall RClick(TObject *Sender)
{
}
```

Delphi

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

Delphi 8 (.NET only)

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

Powe…

```
begin event RClick()

end event RClick
```

VB.NET

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

VB6

```
Private Sub RClick()
End Sub
```

VBA

```
Private Sub RClick()
End Sub
```

VFP

```
LPARAMETERS nop
```

Xbas…

```
PROCEDURE OnRClick(oSplitBar)

RETURN
```

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

Java…

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

VBSc…

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

Visual Data…

```
Procedure OnComRClick
    Forward Send OnComRClick
End_Procedure
```

Visual Objects

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

X++

```
void onEvent_RClick()
{
}
```

XBasic

```
function RClick as v ()
end function
```

dBASE

```
function nativeObject_RClick()
return
```

event Show (Obj as Variant, Visible as Boolean)

Occurs when an object requires to be shown or hidden.

Type	Description
Obj as Variant	A Variant expression that could be one of the following: • String expression that indicates the name of the component • A long expression that specifies the handle of the window • An IUnknown or IDispatch interface that identifies the component (Object for VB, GetOcx() for /COM objects on /NET framework , DefaultInterface for Delphi, nativeObject for dBase, and so on) • An object of Control (System.Windows.Forms) type that specifies the control (/NET assembly, when objects has been added using the AddObjectLT(obj) / AddObjectRB(obj) methods to be shown or hidden. For instance, if the ObjectsLT property is "Command1,Command2", the obj parameter could be "Command1" or "Command2"
Visible as Boolean	A Boolean expression that specifies whether the component requires to be shown (True) or hidden (False).

The split bar control fires the Show event when a component/control/object requires to be shown or hidden. The HideOnLimit property gets or sets a value that indicates whether the splitting objects are hidden when the split bar is closed to its limit. If the HideOnLimit property is True, the control automatically shows or hides the components associated with the split bar when it is close to the limit. The LimitLT property specifies the expression that determines the limit to drag the splitter to left/top side of its container. The LimitRB property specifies the expression that determines the limit to drag the splitter to right/bottom side of its container.

The /COM version may use one of the following to show / hide the object:

- using the Visible property of the extended control. The ExtendedVisible property specifies a list of property names separated by comma character, that indicates the Visible property of the extended control. The Visible property of an extended control shows or hides object. By default, the ExtendedVisible property is "Visible".
- using the ShowWindow API, if the handle of the window can be detected using the

IOleWindow::GetWindow, from obj parameter (VC++ environment)
- using the ShowWindow API, if the obj refers to a handle of the window (Delphi enviroment)

The /NET version shows or hides the objects:

- using the Visible property of the Control object (System.Windows.Forms)

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

C#
```
private void Show(object sender,object   Obj,bool   Visible)
{
}
```

VB
```
Private Sub Show(ByVal sender As System.Object,ByVal Obj As Object,ByVal Visible
As Boolean) Handles Show
End Sub
```

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

C#
```
private void Show(object sender, AxEXSPLITBARLib._ISplitBarEvents_ShowEvent e)
{
}
```

C++
```
void OnShow(VARIANT   Obj,BOOL   Visible)
{
}
```

C++ Builder
```
void __fastcall Show(TObject *Sender,Variant   Obj,VARIANT_BOOL   Visible)
{
}
```

Delphi
```
procedure Show(ASender: TObject; Obj : OleVariant;Visible : WordBool);
begin
end;
```

Delphi 8 (.NET only)
```
procedure Show(sender: System.Object; e:
AxEXSPLITBARLib._ISplitBarEvents_ShowEvent);
begin
end;
```

Powe…
```
begin event Show(any  Obj,boolean  Visible)

end event Show
```

VB.NET
```
Private Sub Show(ByVal sender As System.Object, ByVal e As
AxEXSPLITBARLib._ISplitBarEvents_ShowEvent) Handles Show
End Sub
```

VB6
```
Private Sub Show(ByVal Obj As Variant,ByVal Visible As Boolean)
End Sub
```

VBA
```
Private Sub Show(ByVal Obj As Variant,ByVal Visible As Boolean)
End Sub
```

VFP
```
LPARAMETERS Obj,Visible
```

Xbas…
```
PROCEDURE OnShow(oSplitBar,Obj,Visible)

RETURN
```

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

Java…
```
<SCRIPT EVENT="Show(Obj,Visible)" LANGUAGE="JScript">
</SCRIPT>
```

VBSc…
```
<SCRIPT LANGUAGE="VBScript">
Function Show(Obj,Visible)
End Function
</SCRIPT>
```

Visual Data…
```
Procedure OnComShow Variant   llObj Boolean   llVisible
    Forward Send OnComShow llObj llVisible
End_Procedure
```

Visual Objects
```
METHOD OCX_Show(Obj,Visible) CLASS MainDialog
RETURN NIL
```

X++

```
void onEvent_Show(COMVariant _Obj,boolean _Visible)
{
}
```

XBasic

```
function Show as v (Obj as A,Visible as L)
end function
```

dBASE

```
function nativeObject_Show(Obj,Visible)
return
```