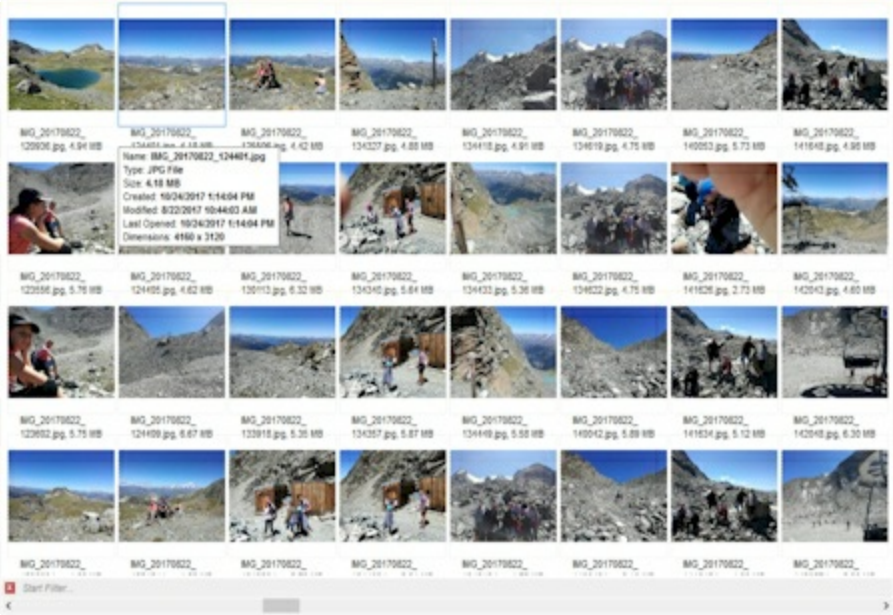
ExThumbnail

Exontrol's exThumbnail component provides thumbnail views for your files. Thumbnails are reduced-size versions of pictures, used to make it easier to scan and recognize them, serving the same role for images as a normal text index does for words.

Features include:

- Multiple-Files, Selection Thumbnail Support
- Filter-Prompt Support, allows you to filter the thumbnails as you type in the control's filter bar
- Sort Support (by name, size, date, dimensions, and so on)
- Context-Menu Support
- Grid/Stack Mode Support
- Auto-Update Support
- Scroll-Bars, AutoDrag Support
- Expression support for Caption, ToolTipText, ToolTipTitle, FormatABC, so you can programmatically displays the caption, tooltip, and so on
- OLE Drag and Drop support.
- Support for all registered graphic formats.
- Ability to save the thumbnail.
- Ability to provide a thumbnail for web pages.
- Ability to resize the thumbnail view.
- BMP, JPG, GIF, PNG, TIFF Support (Ability to save the thumbnail as BMP, JPG, GIF, PNG, TIFF formats)
- WYSWYG Template/Layout Editor support.
- Ability to draw semi-transparent HTML captions on the thumbnail.
- Multiple-Lines HTML Tooltip support.
- ANSI and UNICODE versions available



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 AcceptFoldersEnum

The AcceptFoldersEnum type specifies whether the control supports dropping folders. The AcceptFolders property specifies whether the control accepts drag-and-drop folders. The AcceptFoldersEnum type supports the following values:

Name	Value	Description
exDenyFolderContent	0	The control disables including the folder's content.
exIncludeSubFilesOnly	-1	Includes only the sub-files of the folder.
exIncludeSubFoldersOnly	1	Includes only the sub-folders.
exIncludeAny	2	Includes sub-folders and sub-files.

constants AlignmentEnum

The AlignmentEnum type supports the following values:

Name	Value	Description
LeftAlignment	0	Left Alignment
CenterAlignment	1	Center Alignment
RightAlignment	2	Right Alignment

constants AppearanceEnum

The AppearanceEnum enumeration is used to specify the appearance of the thumbnail control. Use the Appearance property to change the control's appearance.

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

constants AutoDragEnum

The AutoDragEnum type indicates what the control does when the user clicks and start dragging an item. The AutoDrag property indicates the way the component supports the AutoDrag feature.

Name	Value	Description
exAutoDragNone	0	AutoDrag is disabled.
exAutoDragScroll	16	The component scrolls its content by clicking the object and dragging to a new position.
exAutoDragScrollOnRight	1048576	The component is scrolled by clicking the object and dragging to a new position.

constants BackgroundPartEnum

The BackgroundPartEnum type specifies parts of the control, whose background you can change. Use the Background property to specify a background color or a visual appearance for specific parts in the control. The BackgroundPartEnum type supports the following values:

Name	Value	Description
exFooterFilterBarButton	1	Specifies the background color for the closing button in the filter bar.
exThumbnailBackColorAlt	175	Specifies the alternate background color for thumbnail's view.
exThumbnailForeColorAlt	176	Specifies the alternate foreground color for thumbnail's view.
exThumbnailBorderColor	177	Specifies the color to show the thumbnail's border.
exThumbnailSelBorderColor	178	Specifies the color to show the selected thumbnail.
exThumbnailSelBorderColorHide	179	Specifies the color to show the selected thumbnail, while the control has no focus.
exVSUp	256	The up button in normal state.
exVSUpP	257	The up button when it is pressed.
exVSUpD	258	The up button when it is disabled.
exVSUpH	259	The up button when the cursor hovers it.
exVSThumb	260	The thumb part (exThumbPart) in normal state.
exVSThumbP	261	The thumb part (exThumbPart) when it is pressed.
exVSThumbD	262	The thumb part (exThumbPart) when it is disabled.
exVSThumbH	263	The thumb part (exThumbPart) when cursor hovers it.
exVSDown	264	The down button in normal state.
exVSDownP	265	The down button when it is pressed.
exVSDownD	266	The down button when it is disabled.
exVSDownH	267	The down button when the cursor hovers it.
exVSLower	268	The lower part (exLowerBackPart) in normal state.
exVSLowerP	269	The lower part (exLowerBackPart) when it is pressed.

exVSLowerD	270	The lower part (exLowerBackPart) when it is disabled.
exVSLowerH	271	The lower part (exLowerBackPart) when the cursor hovers it.
exVSUpper	272	The upper part (exUpperBackPart) in normal state.
exVSUpperP	273	The upper part (exUpperBackPart) when it is pressed.
exVSUpperD	274	The upper part (exUpperBackPart) when it is disabled.
exVSUpperH	275	The upper part (exUpperBackPart) when the cursor hovers it.
exVSBack	276	The background part (exLowerBackPart and exUpperBackPart) in normal state.
exVSBackP	277	The background part (exLowerBackPart and exUpperBackPart) when it is pressed.
exVSBackD	278	The background part (exLowerBackPart and exUpperBackPart) when it is disabled.
exVSBackH	279	The background part (exLowerBackPart and exUpperBackPart) when the cursor hovers it.
exHSLeft	384	The left button in normal state.
exHSLeftP	385	The left button when it is pressed.
exHSLeftD	386	The left button when it is disabled.
exHSLeftH	387	The left button when the cursor hovers it.
exHSThumb	388	The thumb part (exThumbPart) in normal state.
exHSThumbP	389	The thumb part (exThumbPart) when it is pressed.
exHSThumbD	390	The thumb part (exThumbPart) when it is disabled.
exHSThumbH	391	The thumb part (exThumbPart) when the cursor hovers it.
exHSRight	392	The right button in normal state.
exHSRightP	393	The right button when it is pressed.
exHSRightD	394	The right button when it is disabled.
exHSRightH	395	The right button when the cursor hovers it.
exHSLower	396	The lower part (exLowerBackPart) in normal state.

exHSLowerP	397	The lower part (exLowerBackPart) when it is pressed.
exHSLowerD	398	The lower part (exLowerBackPart) when it is disabled.
exHSLowerH	399	The lower part (exLowerBackPart) when the cursor hovers it.
exHSUpper	400	The upper part (exUpperBackPart) in normal state.
exHSUpperP	401	The upper part (exUpperBackPart) when it is pressed.
exHSUpperD	402	The upper part (exUpperBackPart) when it is disabled.
exHSUpperH	403	The upper part (exUpperBackPart) when the cursor hovers it.
exHSBack	404	The background part (exLowerBackPart and exUpperBackPart) in normal state.
exHSBackP	405	The background part (exLowerBackPart and exUpperBackPart) when it is pressed.
exHSBackD	406	The background part (exLowerBackPart and exUpperBackPart) when it is disabled.
exHSBackH	407	The background part (exLowerBackPart and exUpperBackPart) when the cursor hovers it.
exSBtn	324	All button parts (L1-L5, LButton, exThumbPart, RButton, R1-R6), in normal state.
exSBtnP	325	All button parts (L1-L5, LButton, exThumbPart, RButton, R1-R6), when it is pressed.
exSBtnD	326	All button parts (L1-L5, LButton, exThumbPart, RButton, R1-R6), when it is disabled.
exSBtnH	327	All button parts (L1-L5, LButton, exThumbPart, RButton, R1-R6), when the cursor hovers it .
exVSThumbExt	503	The thumb-extension part in normal state.
exVSThumbExtP	504	The thumb-extension part when it is pressed.
exVSThumbExtD	505	The thumb-extension part when it is disabled.
exVSThumbExtH	506	The thumb-extension when the cursor hovers it.
exHSThumbExt	507	The thumb-extension in normal state.
exHSThumbExtP	508	The thumb-extension when it is pressed.

exHSThumbExtD	509	The thumb-extension when it is disabled.
exHSThumbExtH	510	The thumb-extension when the cursor hovers it.
exScrollSizeGrip	511	Specifies the visual appearance of the control's size grip when both scrollbars are shown.

constants exClipboardFormatEnum

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

Name	Value	Description
exCFText	1	Null-terminated, plain ANSI text in a global memory bloc
exCFBitmap	2	A bitmap compatible with Windows 2.X
exCFMetafile	3	A Windows metafile with some additional information about how the metafile should be displayed
exCFDIB	8	A global memory block containing a Windows device-independent bitmap (DIB)
exCFPalette	9	A color-palette handle
exCFEMetafile	14	A Windows enhanced metafile
exCFFiles	15	A collection of files. Use Files property to get the collection of files
exCFRTF	-16639	A RTF document

constants exOLEDragOverEnum

State transition constants for the OLEDragOver event

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

constants exOLEDropEffectEnum

Drop effect constants for OLE drag and drop events.

Name	Value	Description
exOLEDropEffectNone	0	Drop target cannot accept the data, or the drop operation was cancelled.
exOLEDropEffectCopy	1	Drop results in a copy of data from the source to the target. The original data is unaltered by the drag operation.
exOLEDropEffectMove	2	Drop results in data being moved from drag source to drop source. The drag source should remove the data from itself after the move.
exOLEDropEffectScroll	-2147483648	This one is not implemented.

constants exOLEDropModeEnum

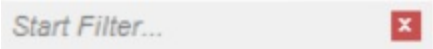
Constants for the OLEDropMode property, that defines how the control accepts OLE drag and drop operations. Use the OLEDropMode property to set how the component handles drop operations.

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

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

constants FilterBarVisibleEnum

The FilterBarPromptVisible property , specifies how the control's filter bar is displayed and behave. The FilterBarVisibleEnum type includes several flags that can be combined together, as described bellow:

Name	Value	Description
exFilterBarHidden	0	No filter bar is shown while there is no filter applied. The control's filter bar is automatically displayed as soon a a filter is applied.
exFilterBarVisible	1	Specifies that the control's filter bar is visible.
exFilterBarToggle	256	The exFilterBarToggle flag specifies that the user can close the control's filter bar (removes the control's filter) by clicking the close button of the filter bar or by pressing the CTRL + F, while the control's filter bar is visible. If no filter bar is displayed, the user can display the control's filter bar by pressing the CTRL + F key. While the control's filter bar is visible the user can navigate though the list or control's filter bar using the ALT + Up/Down keys. If missing, the control's filter bar is always shown if any of the following flags is present exFilterBarPromptVisible, exFilterBarVisible, exFilterBarCaptionVisible.
exFilterBarShowCloseIfRequired	512	The exFilterBarShowCloseIfRequired flag indicates that the close button of the control's filter bar is displayed only if the control has any currently filter applied. The Background(exFooterFilterBarButton) property on -1 hides permanently the close button of the control's filter bar. 
exFilterBarShowCloseOnRight	1024	The exFilterBarShowCloseOnRight flag specifies that the close button of the control's filter bar should be displayed on the right side. 

constants FilterPromptEnum

The FilterPromptEnum type specifies the type of prompt filtering. Use the FilterBarPromptType property to specify the type of filtering when using the prompt. The FilterBarPromptPattern property specifies the pattern for filtering. The pattern may contain one or more words being delimited by space characters.

The filter prompt feature supports the following values:

Name	Value	Description
exFilterPromptContainsAll	1	The list includes the items that contains all specified sequences in the filter. Can be combined with exFilterPromptCaseSensitive, exFilterPromptStartWords, exFilterPromptEndWords or exFilterPromptWords
exFilterPromptContainsAny	2	The list includes the items that contains any of specified sequences in the filter. Can be combined with exFilterPromptCaseSensitive, exFilterPromptStartWords, exFilterPromptEndWords or exFilterPromptWords
exFilterPromptStartWith	3	The list includes the items that starts with any specified sequences in the filter. Can be combined with exFilterPromptCaseSensitive, exFilterPromptStartWords, exFilterPromptEndWords or exFilterPromptWords
exFilterPromptEndWith	4	The list includes the items that ends with any specified sequences in the filter. Can be combined with exFilterPromptCaseSensitive, exFilterPromptStartWords, exFilterPromptEndWords or exFilterPromptWords
exFilterPromptPattern	16	The filter indicates a pattern that may include wild characters to be used to filter the items in the list. The FilterBarPromptPattern property may include wild characters as follows: • '?' for any single character • '*' for zero or more occurrences of any character • '#' for any digit character • ' ' space delimits the patterns inside the filter

exFilterPromptCaseSensitive	256	Filtering the list is case sensitive. Can be combined with exFilterPromptContainsAll, exFilterPromptContainsAny, exFilterPromptStartWith or exFilterPromptEndWith.
exFilterPromptStartWords	4608	The list includes the items that starts with specified words, in any position. Can be combined with exFilterPromptContainsAll, exFilterPromptContainsAny, exFilterPromptStartWith or exFilterPromptEndWith.
exFilterPromptEndWords	8704	The list includes the items that ends with specified words, in any position. Can be combined with exFilterPromptContainsAll, exFilterPromptContainsAny, exFilterPromptStartWith or exFilterPromptEndWith.
exFilterPromptWords	12800	The filter indicates a list of words. Can be combined with exFilterPromptContainsAll, exFilterPromptContainsAny, exFilterPromptStartWith or exFilterPromptEndWith.

constants HTMLRotateEnum

The HTMLRotateEnum expression specifies how the control displays the HTML caption. The CaptionRotate property specifies whether the caption is displayed rotated. The HTMLRotateEnum expression supports the following values:

Name	Value	Description
exHTMLHorizontal	0	Specifies that the HTML text is horizontally displayed. This flag can be combined with exHTMLMirror flag.
exHTMLVertical	1	Specifies that the HTML text is vertically displayed. This flag can be combined with exHTMLMirror flag.
exHTMLMirror	16	Specifies that the HTML text is displayed in mirror. This flag can be combined with exHTMLHorizontal or exHTMLVertical flag.

constants PictureDisplayEnum

Specifies how the picture is displayed on the control's background. Use the PictureDisplay property to specify how the control displays its picture (on the background). The Picture property displays specified picture on the control's background, while the Thumbnail property specifies the thumbnail view for InputFile property.

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

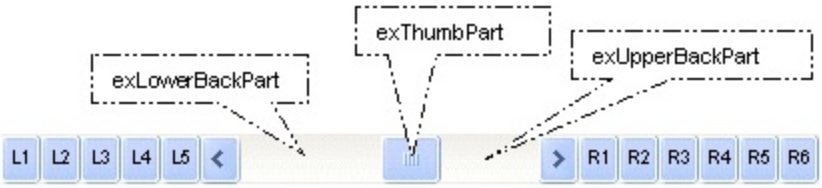
constants ScrollBarEnum

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

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

constants ScrollPartEnum

The ScrollPartEnum type defines the parts in the control's scrollbar. Use the ScrollPartVisible property to specify the visible parts in the control's scroll bar. Use the ScrollPartCaption property to specify the caption being displayed in any part of the control's scrollbar.



Name	Value	Description
exExtentThumbPart	65536	The thumb-extension part.
exLeftB1Part	32768	(L1) The first additional button, in the left or top area. By default, this button is hidden.
exLeftB2Part	16384	(L2) The second additional button, in the left or top area. By default, this button is hidden.
exLeftB3Part	8192	(L3) The third additional button, in the left or top area. By default, this button is hidden.
exLeftB4Part	4096	(L4) The forth additional button, in the left or top area. By default, this button is hidden.
exLeftB5Part	2048	(L5) The fifth additional button, in the left or top area. By default, this button is hidden.
exLeftBPart	1024	(<) The left or top button. By default, this button is visible.
exLowerBackPart	512	The area between the left/top button and the thumb. By default, this part is visible.
exThumbPart	256	The thumb part or the scroll box region. By default, the thumb is visible.
exUpperBackPart	128	The area between the thumb and the right/bottom button. By default, this part is visible.
exBackgroundPart	640	The union between the exLowerBackPart and the exUpperBackPart parts. By default, this part is visible.
exRightBPart	64	(>) The right or down button. By default, this button is visible.
		(R1) The first additional button in the right or down

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

constants StateChangeEnum

The StateChangeEnum type specifies different control's state. The StateChange event occurs once the control's state is changed. The StateChangeEnum type supports the following values:

Name	Value	Description
SelChangeState	3	Fired when the control's selection has been changed.
ShowContextMenu	20	Occurs when the control is about to display the object's context menu.
ExecuteContextMenu	21	Occurs when the control is about to execute a command from the object's context menu.

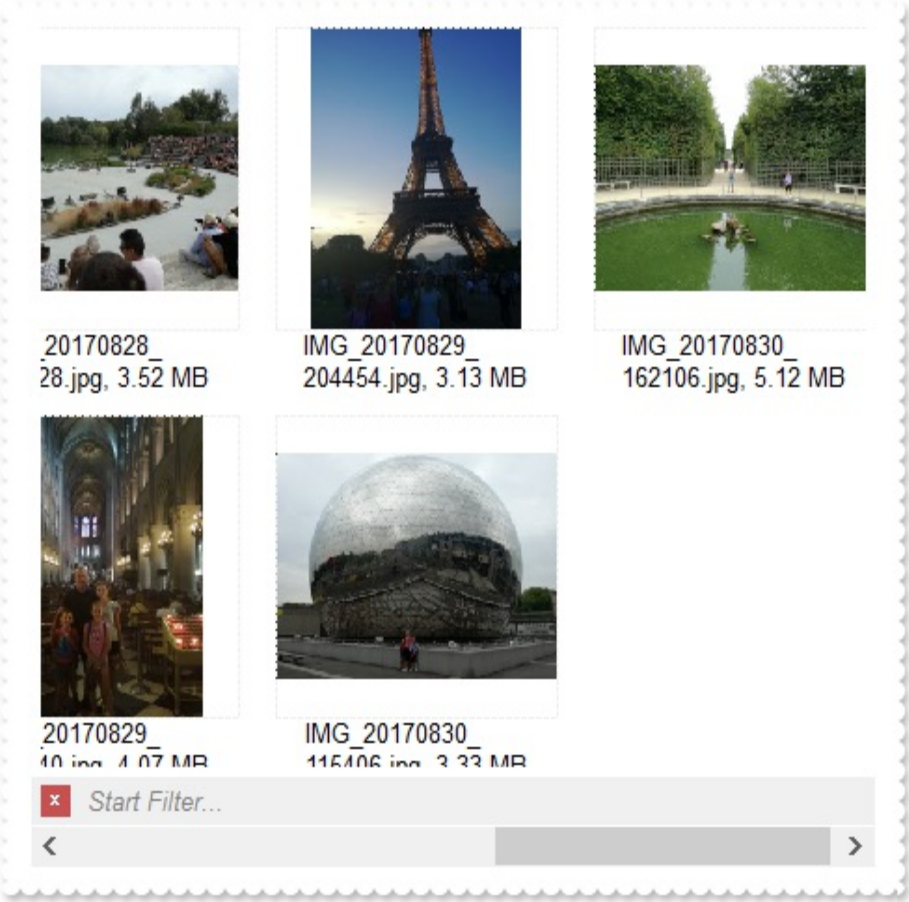
constants TextAlignEnum

Specifies the alignment of the caption in the thumbnail control. Use the Alignment property to specify the caption's alignment.

Name	Value	Description
exAlignTopLeft	0	The object or text is aligned in the top-left side of the control element.
exAlignTopCenter	1	The object or text is aligned in the top-center side of the control element.
exAlignTopRight	2	The object or text is aligned in the top-right side of the control element.
exAlignMiddleLeft	16	The object or text is aligned in the middle-left side of the control element.
exAlignMiddleCenter	17	The object or text is aligned in the center of the control element.
exAlignMiddleRight	18	The object or text is aligned in the middle-right side of the control element.
exAlignBottomLeft	32	The object or text is aligned in the bottom-left side of the control element.
exAlignBottomCenter	33	The object or text is aligned in the bottom-center side of the control element.
exAlignBottomRight	34	The object or text is aligned in the bottom-right side of the control element.

constants ThumbnailModeEnum

The ThumbnailModeEnum type specifies the type of modes the control supports. The Mode property changes the thumbnail's mode. The ThumbnailModeEnum type supports the following flags:

Name	Value	Description
exThumbnailGrid	0	The thumbnails are arranged on rectangles. 
		Indicates that thumbnails are displayed as a stack.

exThumbnailStack	1	
exThumbnailAutoFit	256	Specifies the thumbnail to fit the frame's client area.
exThumbnailStretch	512	Stretches the thumbnail on the frame's client area.
exThumbnailCenter	1024	Centers the thumbnails.
exThumbnailRight	2048	Right aligns the thumbnails.
exThumbnailAllowResize	4096	Specifies whether the user can resize at runtime the thumbnail view by dragging the view while clicking the middle-mouse.
exThumbnailAutoFitOnDblClk	8192	Performs the auto-fit layout when the user double clicks the control's client area.
exThumbnailKeepAspectRatio	16384	The thumbnail is keeping its aspect ratio. The ThumbnailWidth / ThumbnailHeight property specifies the size to display the thumbnails.
exThumbnailBrowseOnDblClk	32768	Browses for a new folder once the user double clicks it. The AcceptFolders property should be exIncludeSubFoldersOnly or exIncludeAny, so folders are being shown within the control's content. For instance, the Mode = exThumbnailBrowseOnDblClk simulates an Explorer (Files and Folders browser) control. The user can browse back for the parent-folder using the

Backspace key.

constants ThumbnailSortEnum

The ThumbnailSortEnum type defines the ways the control can sort the files. You can use the Sort property to programmatically sort the files within the control. The ThumbnailSortEnum type supports the following values:

Name	Value	Description
exThumbailUnsorted	0	The files are being shown as they were loaded. No sort is performed.
exThumbailSortByFile	1	The files are being arranged based on their full name.
exThumbailSortByName	2	The files are being arranged based on their name.
exThumbailSortByType	3	The files are being arranged based on their type.
exThumbailSortBySize	4	The files are being arranged based on their size.
exThumbailSortByCreated	5	The files are being arranged based on their creation date.
exThumbailSortByModified	6	The files are being arranged based on their modified date.
exThumbailSortByOpened	7	The files are being arranged based on their access date.
exThumbailSortByDimension	8	The files are being arranged based on their dimensions. (Applies to Picture files, slower)
exThumbailSortReverse	256	This flag can be combined with any other not-zero flag to specify a reverse sort order.
exThumbailSortInsensitive	512	The files are sort case insensitive. If the exThumbailSortInsensitive flag is missing the sort is case sensitive

constants ThumbnailTypeEnum

The ThumbnailTypeEnum type specifies the thumbnail being displayed. Use the ThumbnailType property to specify the thumbnail being loaded.

Name	Value	Description
exNoThumbnail	0	No thumbnail is displayed. The Thumbnail property is empty.
exThumbnailBitmap	1	The thumbnail type is a bitmap.
exThumbnailIcon	3	The thumbnail type is an icon.
exThumbnailPicture	9	The thumbnail type is a bitmap, but the original file is a picture being loaded directly.
exThumbnailAvail	255	Specifies whether the control looks for available views such as Picture, Bitmap, and Icon representation. For instance, if the InputFile property points to a picture file, the control loads directly the file, and the ThumbnailType property retrieves the exThumbnailPicture. If the InputFile is not a picture file, but found a bitmap representation, that the ThumbnailType property retrieves the exThumbnailBitmap, and last it gets the exThumbnailIcon.

ExDataObject object

Defines the object that contains OLE drag and drop information.

Name	Description
Clear	Deletes the contents of the ExDataObject object.
Files	Returns an ExDataObjectFiles collection, which in turn contains a list of all filenames used by an ExDataObject object.
GetData	Returns data from an ExDataObject object in the form of a variant.
GetFormat	Returns a value indicating whether an item in the ExDataObject object matches a specified format.
SetData	Inserts data into an ExDataObject object using the specified data format.

method ExDataObject.Clear ()

Deletes the contents of the DataObject object.

Type	Description

The Clear method can be called only for drag sources. The OleDragDrop event notifies your application that the user drags some data on the control.

property ExDataObject.Files as ExDataObjectFiles

Returns a DataObjectFiles collection, which in turn contains a list of all filenames used by a DataObject object.

Type	Description
ExDataObjectFiles	An ExDataObjectFiles object that contains a list of filenames used in OLE drag and drop operations

The Files property is valid only if the format of the clipboard data is exCFFiles. The OleDragDrop event notifies your application that the user drags some data on the control.

method ExDataObject.GetData (Format as Integer)

Returns data from a DataObject object in the form of a variant.

Type	Description
Format as Integer	An exClipboardFormatEnum expression that defines the data's format

Return	Description
Variant	A Variant value that contains the ExDataObject's data in the given format

Use GetData property to retrieve the clipboard's data that has been dragged to the control. It's possible for the GetData and SetData methods to use data formats other than exClipboardFormatEnum , including user-defined formats registered with Windows via the RegisterClipboardFormat() API function. The GetData method always returns data in a byte array when it is in a format that it is not recognized. Use the Files property to retrieves the filenames if the format of data is exCFFiles

method ExDataObject.GetFormat (Format as Integer)

Returns a value indicating whether the ExDataObject's data is of the specified format.

Type	Description
Format as Integer	A constant or value that specifies a clipboard data format like described in exClipboardFormatEnum enum.

Return	Description
Boolean	A boolean value that indicates whether the ExDataObject's data is of specified format.

Use the GetFormat property to verify if the ExDataObject's data is of a specified clipboard format. The GetFormat property retrieves True, if the ExDataObject's data format matches the given data format.

method ExDataObject.SetData ([Value as Variant], [Format as Variant])

Inserts data into a ExDataObject object using the specified data format.

Type	Description
Value as Variant	A data that is going to be inserted to ExDataObject object.
Format as Variant	A constant or value that specifies the data format, as described in exClipboardFormatEnum enum

Use SetData property to insert data for OLE drag and drop operations. Use the Files property is you are going to add new files to the clipboard data. The OleDragDrop event notifies your application that the user drags some data on the control.

ExDataObjectFiles object

The ExDataObjectFiles contains a collection of filenames. The ExDataObjectFiles object is used in OLE Drag and drop events. In order to get the list of files used in drag and drop operations you have to use the Files property.

Name	Description
Add	Adds a filename to the Files collection
Clear	Removes all file names in the collection.
Count	Returns the number of file names in the collection.
Item	Returns an specific file name.
Remove	Removes an specific file name.

method ExDataObjectFiles.Add (FileName as String)

Adds a filename to the Files collection

Type	Description
FileName as String	A string expression that indicates a filename.

Use Add method to add your files to ExDataObject object. The OleStartDrag event notifies your application that the user starts dragging items.

method ExDataObjectFiles.Clear ()

Removes all file names in the collection.

Type	Description

Use the Clear method to remove all filenames from the collection.

property ExDataObjectFiles.Count as Long

Returns the number of file names in the collection.

Type	Description
Long	A long value that indicates the count of elements into collection.

You can use "for each" statements if you are going to enumerate the elements into ExDataObjectFiles collection.

property ExDataObjectFiles.Item (Index as Long) as String

Returns a specific file name given its index.

Type	Description
Index as Long	A long expression that indicates the filename's index
String	A string value that indicates the filename

method ExDataObjectFiles.Remove (Index as Long)

Removes a specific file name given its index into collection.

Type	Description
Index as Long	A long expression that indicates the index of filename into collection.

Use Clear method to remove all filenames.

Thumbnail 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: {44BA596F-E225-476B-81B1-3BDE56AC595A}. The object's program identifier is: "Exontrol.Thumbnail". The /COM object module is: "ExThumbnail.dll"

Exontrol's exThumbnail component provides thumbnail views for your files. Thumbnails are reduced-size versions of pictures, used to make it easier to scan and recognize them, serving the same role for images as a normal text index does for words. The Thumbnail object supports the following properties and methods:

Name	Description
AcceptFiles	Specifies whether the control accepts drag-and-drop files.
AcceptFolders	Specifies whether the control accepts drag-and-drop folders.
AddInputFiles	Adds files to be thumbnailed.
Alignment	Retrieves or sets the alignment of the control's caption.
AllowContextMenu	Gets or sets a value that specifies whether the shell's context menu is shown once the user right clicks the thumbnail.
AnchorFromPoint	Retrieves the identifier of the anchor from point.
Appearance	Retrieves or sets the control's appearance.
AttachTemplate	Attaches a script to the current object, including the events, from a string, file, a safe array of bytes.
AutoDrag	Specifies whether the control automatically scrolls its content as soon as user clicks it.
AutoFit	Layouts the thumbnail views, so all fit the control's client area.
AutoUpdate	Determines whether the control automatically updates the thumbnail's view being changed externally.
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 performing one at a time. This method prevents the control from painting until the EndUpdate method is called.
Bitmap	Retrieves the thumbnail view as a bitmap object.
Borders	Specifies the size of the borders when the thumbnail is

	displayed.
Caption	Indicates the expression to generate the HTML caption to be displayed on the thumbnail.
CaptionRotate	Rotates the HTML caption.
Debug	Displays debug information.
Enabled	Enables or disables the control.
EndUpdate	Resumes painting the control after painting is suspended by the BeginUpdate method.
EventParam	Retrieves or sets a value that indicates the current's event parameter.
ExecuteContextCommand	Executes a context menu command.
ExecuteContextMenu	Executes a command from the object's context menu.
ExecuteTemplate	Executes a template and returns the result.
ExtractFlag	Specifies how the thumbnail is extracted.
ExtractMethod	Specifies the order and the methods the control uses to extract the thumbnails.
ExtractTime	Specifies the time ellaspsed to extract from last object.
FilterBarBackColor	Specifies the background color of the control's filter bar.
FilterBarFor	Specifies the expression that determines whether the thumbnail is included in the filter.
FilterBarForeColor	Specifies the foreground color of the control's filter bar.
FilterBarPrompt	Specifies the HTML caption to be displayed when the filter pattern is missing.
FilterBarPromptPattern	Applies the giving filter/pattern to the control.
FilterBarPromptType	Specifies the type of the filter prompt.
FilterBarVisible	Specifies whether the control's filter bar is visible or hidden.
Font	Retrieves or sets the control's font.
ForeColor	Specifies the control's foreground color.
FormatABC	Formats the A,B,C values based on the giving expression and returns the result.
FormatAnchor	Specifies the visual effect for anchor elements in HTML captions.
FreezeEvents	Prevents the control to fire any event.

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..
InputFile	Specifies the file to display the thumbnail.
InputFiles	Indicates the list of files to be thumbnailed.
KeepOriginalThumbnail	Specifies whether the thumbnail is shown as it is provided.
LimitInputFiles	Limits the number of files the control can display.
Margins	Specifies the distance between thumbnails, when multiple thumbnails are displayed on the control.
Mode	Specifies how thumbnails are arranged on the control's client area.
OLEDrag	Causes a component to initiate an OLE drag/drop operation.
OLEDropMode	Returns or sets how a target component handles drop operations
OutputFile	Specifies the file where to save the thumbnail.
Padding	Generates space around thumbnail. Clears the area around the content (inside the border) of the thumbnail.
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	Refreses the control.
ReplaceIcon	Adds a new icon, replaces an icon or clears the control's image list.
SaveAs	Save the thumbnail, as a picture file in specified format giving by the extension (characters after last dot, determines the graphical/ format of the file).
ScrollButtonHeight	Specifies the height of the button in the vertical scrollbar.
ScrollButtonWidth	Specifies the width of the button in the horizontal scrollbar.
ScrollFont	Retrieves or sets the scrollbar's font.
ScrollHeight	Specifies the height of the horizontal scrollbar.
ScrollOrderParts	Specifies the order of the buttons in the scroll bar.

ScrollPartCaption	Specifies the caption being displayed on the specified scroll part.
ScrollPartCaptionAlignment	Specifies the alignment of the caption in the part of the scroll bar.
ScrollPartEnable	Indicates whether the specified scroll part is enabled or disabled.
ScrollPartVisible	Indicates whether the specified scroll part is visible or hidden.
ScrollThumbSize	Specifies the size of the thumb in the scrollbar.
ScrollToolTip	Specifies the tooltip being shown when the user moves the scroll box.
ScrollWidth	Specifies the width of the vertical scrollbar.
Select	Select property indicates the name of the selected thumbnail.
ShowContextMenu	Specifies the object's context menu.
ShowImageList	Specifies whether the control's image list window is visible or hidden.
ShowToolTip	Shows the specified tooltip at given position.
SingleCaption	Indicates the expression to generate the HTML caption to be displayed on the thumbnail, when the control shows a single thumbnail.
SingleSel	Retrieves or sets a value that indicates whether the control supports single or multiple selection.
Sort	Sorts the thumbnails.
StatusCaption	Indicates the expression to generate the HTML caption to be displayed on the thumbnail's status bar.
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.
Thumbnail	Retrieves the thumbnail view of the file.
ThumbnailFromPoint	Retrieves the thumbnail from point.
ThumbnailHeight	Specifies the height to display the thumbnails.

ThumbnailMaxHeight	Specifies the maximum height to display the thumbnails.
ThumbnailMaxWidth	Specifies the maximum width to display the thumbnails.
ThumbnailMinHeight	Specifies the minimum height to display the thumbnails.
ThumbnailMinWidth	Specifies the minimum width to display the thumbnails.
ThumbnailType	Retrieves the type of the thumbnail view.
ThumbnailWidth	Specifies the width to display the thumbnails.
Timeout	Specifies a value that indicates the number of milliseconds to extract the object's thumbnail.
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	Gets or sets the tooltip's expression associated with the thumbnail control.
ToolTipTitle	Gets or sets the tooltip title associated with the thumbnail control.
ToolTipWidth	Specifies a value that indicates the width of the tooltip window, in pixels.
Transparency	Specifies the transparency to display the text in the control.
Version	Retrieves the control's version.
VirtualMode	Specifies whether the control is running in virtual mode.
WordWrap	Indicates whether a multiline label control automatically wraps words to the beginning of the next line when necessary.

property Thumbnail.AcceptFiles as Boolean

Specifies whether the control accepts drag-and-drop files.

Type	Description
Boolean	A boolean expression that specifies whether the control accepts drag-and-drop files.

By default, the AcceptFiles property is True. Use the AcceptFiles property to disable drag-and-drop files in the thumbnail control. The AcceptFolders property specifies whether the control accepts drag-and-drop folders. Use the Thumbnail property to retrieve the thumbnail picture. Use the OutputFile property to save the thumbnail picture. The Change event notifies your application when the user drags a file into the thumbnaill control.

You can add support for OLE Drag and Drop operations using the following events:

- OLEStartDrag event.
- OLEDragDrop event

You can use any of the following properties/method to add files into the control:

- InputFile property specifies the file or the web page to display its thumbnail.
- InputFiles property indicates the list of files to be thumbnailed
- AddInputFiles method adds files to be thumbnailed.

property Thumbnail.AcceptFolders as AcceptFoldersEnum

Specifies whether the control accepts drag-and-drop folders.

Type	Description
AcceptFoldersEnum	A AcceptFoldersEnum expression that specifies whether the control accepts drag-and-drop folders.

By default, the AcceptFolders property is exDenyFolderContent, which denies dropping any folder. You can use the AcceptFolders property to allow user drop folders or files from a folders. The AcceptFiles property specifies whether the control accepts drag-and-drop files.

method Thumbnail.AddInputFiles (Files as Variant)

Adds files to be thumbnailed.

Type	Description
Files as Variant	A string expression that specifies a list of files to be added (separated by \r\n sequence) or a safe array of variant, where each element could be a list of files to be added (separated by \r\n sequence)

The AddInputFiles method adds new files to the control. You can clear the files into the control by calling the InputFile property on "". The control fires the Changing event before updating the thumbnail view, and the Change event once the thumbnail is changed. Use the ThumbnailType property to specify whether the thumbnail is a bitmap or an icon. Use the AcceptFiles property to disable drag-and-drop files in the thumbnail control. The AcceptFolders property specifies whether the control accepts drag-and-drop folders. The LimitInputFiles property limits the number of files the control can display.

You can use any of the following properties/method to add files into the control:

- InputFile property specifies the file or the web page to display its thumbnail.
- InputFiles property indicates the list of files to be thumbnailed
- AddInputFiles method adds files to be thumbnailed.

property Thumbnail.Alignment as TextAlignEnum

Retrieves or sets the alignment of the control's caption.

Type	Description
TextAlignEnum	A TextAlignEnum expression that specifies the alignment of the caption into the thumbnail control.

By default, the Alignment property is exAlignMiddleCenter. Use the Alignment property to align the caption in the thumbnail. Use the Caption property to display a HTML caption in the thumbnail. The SingleCaption property indicates the expression to generate the HTML caption to be displayed on the thumbnail, when the control shows a single thumbnail (for instance you filter and you got a single result). Use the Borders property to specify the size of the margins where the thumbnail picture is displayed.

property Thumbnail.AllowContextMenu as Boolean

Enables or disables the file's context menu.

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

By default, the AllowContextMenu property is True. Use the AllowContextMenu to disable the control's context menu. The control's context menu is displayed when the user does a right click on the file or the folder. The system controls the items being inserted to the control's context menu. Use the ExecuteContextCommand method to execute a command from the file's context menu. The ShowContextMenu property indicates the items to be displayed on the object's context menu. The ShowContextMenu property has effect only during the StateChange event, when the State parameter is ShowContextMenu. The ShowContextMenu property can be used to disable, update, remove or add new items. The ExecuteContextMenu property specifies the identifier of the command to be executed (id option in the ShowContextMenu property). The ExecuteContextMenu property has effect only during the StateChange event, when the State parameter is ExecuteContextMenu. The Background(exThumbnailSelBorderColor) property specifies the color to show the thumbnail's border/frame when focused. By default, the Background(exThumbnailSelBorderColor) property is 0x8000000D. You can set on 0 or -1, to prevent showing the thumbnail's selected frame

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

Retrieves the identifier of the anchor from point.

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

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

The following VB sample displays (as tooltip) the identifier of the anchor element from the cursor:

```
Private Sub Thumbnail1_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
   With Thumbnail1
      .ShowToolTip .AnchorFromPoint(-1, -1)
   End With
End Sub
```

The following VB.NET sample displays (as tooltip) the identifier of the anchor element from the cursor:

```
Private Sub AxThumbnail1_MouseMoveEvent(ByVal sender As System.Object, ByVal e As AxEXTHUMBNAILLib._IThumbnailEvents_MouseMoveEvent) Handles AxThumbnail1.MouseMoveEvent
   With AxThumbnail1
      .ShowToolTip(.get_AnchorFromPoint(-1, -1))
   End With
```

```
End Sub
```

The following C# sample displays (as tooltip) the identifier of the anchor element from the cursor:

```
private void axThumbnail1_MouseMoveEvent(object sender,
AxEXTHUMBNAILLib._IThumbnailEvents_MouseMoveEvent e)
{
   axThumbnail1.ShowToolTip(axThumbnail1.get_AnchorFromPoint(-1, -1));
}
```

The following C++ sample displays (as tooltip) the identifier of the anchor element from the cursor:

```
void OnMouseMoveThumbnail1(short Button, short Shift, long X, long Y)
{
   COleVariant vtEmpty; V_VT( &vtEmpty ) = VT_ERROR;
   m_thumbnail.ShowToolTip( m_thumbnail.GetAnchorFromPoint( -1, -1 ), vtEmpty,
vtEmpty, vtEmpty );
}
```

The following VFP sample displays (as tooltip) the identifier of the anchor element from the cursor:

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

with thisform
   With .Thumbnail1
      .ShowToolTip(.AnchorFromPoint(-1, -1))
   EndWith
endwith
```

property Thumbnail.Appearance as AppearanceEnum

Retrieves or sets the control's appearance.

Type	Description
AppearanceEnum	An AppearanceEnum expression that specifies the control's appearance.

By default, the thumbnail control displays no borders. Use the Appearance property to change the borders of the thumbnail control. Use the Borders property to specify the size of the margins where the thumbnail picture is displayed. Use the Picture property to display a picture on the control's background. Use the PictureDisplay property to layout the control's Picture on the background. Use the BackColor property to specify the control's background color. Use the KeepOriginalThumbnail property to specify whether the thumbnail displays its content using a transparent color.

method Thumbnail.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 Thumbnail1_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 Thumbnail.AutoDrag as AutoDragEnum

Specifies whether the control automatically scrolls its content as soon as user clicks it.

Type	Description
AutoDragEnum	An AutoDragEnum expression that specifies whether the control automatically scrolls its content as soon as user clicks it.

By default, the AutoDrag property is exAutoDragScroll, which indicates that the control's content is scrolled as soon as the user clicks and drags the cursor.

method Thumbnail.AutoFit ()

Layouts the thumbnail views, so all fit the control's client area.

Type	Description

property Thumbnail.AutoUpdate as Boolean

Determines whether the control automatically updates the thumbnail's view being changed externally.

Type	Description
Boolean	A Boolean expression that determines whether the control automatically updates the thumbnail's view being changed externally.

By default, the AutoUpdate property is False. The AutoUpdate property indicates whether the control automatically updates the thumbnail's view being changed externally. Once the AutoUpdate property is True, the control automatically regenerates the thumbnails for files being changed in the current view. The VirtualMode property specifies whether the control is running in virtual mode.

property Thumbnail.BackColor as Color

Specifies the control's background color.

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

By default, the BackColor property is the system window background color. Use the BackColor property to change the control's background color. Use the KeepOriginalThumbnail property to specify whether the thumbnail is displayed using a transparent color. Use the <bgcolor> HTML tag to display portion of text using a different background color using the Caption property. Use the Borders property to specify the size of the margins where the thumbnail picture is displayed. Use the Picture property to display a picture on the control's background. Use the PictureDisplay property to layout the control's Picture on the background.

property Thumbnail.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 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.

method Thumbnail.BeginUpdate ()

Maintains performance when multiple changes are performing one at a time. his method prevents the control from painting until the EndUpdate method is called.

Type	Description

This method prevents the control from painting until the EndUpdate method is called. Once that BeginUpdate method was called, you have to make sure that EndUpdate method will be called too.

property Thumbnail.Bitmap as IPictureDisp

Retrieves the thumbnail view as a bitmap object.

Type	Description
IPictureDisp	A Picture/Image object that shows the file's thumbnail view.

The Bitmap property always returns a BITMAP object that shows the file's thumbnail. This property is similar with the Thumbnail property excepts that the last it may returns an ICO, BITMAP, JPEG, TIFF pictures. Use the InputFile property to specify the file whose thumbnail is retrieved. Use the ThumbnailType property to specify whether the thumbnail is a bitmap or an icon. For instance, if you set the ThumbnailType property on exThumbnailIcon, and calling the InputFile property, the control gets the icon representation of the file. If the ThumbnailType property on exThumbnailIcon is exThumbnailBitmap the control tries to get a bitmap representation of the file if it is available, else the Thumbnail property is nothing, and the ThumbnailType property is exNoThumbnail. Use the OutputFile property to save the thumbnail picture.

property Thumbnail.Borders as String

Specifies the size of the borders when the thumbnail is displayed.

Type	Description
String	A String expression that specifies a list of size for margins where the thumbnail picture is displayed. The first number indicates the size of the left margin, the second number indicates the size of the top margin, the third number indicates the size of the right margin, and the last number indicates the size of the bottom margin. All these numbers indicate pixels. The list is separated by space characters.

By default, the Borders property is "4 4 4 4". Use the Alignment property to specify the caption being displayed on the thumbnail control. Use the Transparency property to specify the percent of transparency being used to paint the caption on the thumbnail. Use the Appearance property to specify whether the thumbnail displays a border.

You can use the following properties to define the distance and padding of the thumbnails

- The Borders property indicates the margins where the thumbnail is displayed, relative to the control's border.
- The Margins property specifies the distance between thumbnails, when multiple thumbnails are displayed on the control.
- The Padding property generates space around thumbnail. Clears the area around the content (inside the border) of the thumbnail.

property Thumbnail.Caption as String

Specifies the control's caption.

Type	Description
String	A String value that defines the expression to generate the HTML caption to be displayed on the thumbnail.

By default, the Caption property is "ffolder ? (fname + (len(0:=fsizeF) ? `, ` + =:0 : ``)) : ffile", which displays name and size for a file, the name for a folder, or the full name if the object is no file or folder. Use the Caption property to specify a caption to be displayed on the thumbnail. The SingleCaption property indicates the expression to generate the HTML caption to be displayed on the thumbnail, when the control shows a single thumbnail (for instance you filter and you got a single result). Use the Alignment property to specify the caption's alignment. The CaptionRotate property rotates the HTML caption. Use the Transparency property to specify the transparency to display the caption on the thumbnail control. Use the Font property to specify the font to display the caption on the thumbnail. Use the ForeColor property to specify the control's foreground color. Use the WordWrap property to wrap the caption on the control. The StatusCaption property indicates the expression to generate the HTML caption to be displayed on the thumbnail's status bar.

For instance:

- "ffile" displays the full name of the file
- "`Name: <b>` + fname", displays just the name of the file

- "`Created: <b>` + longdate(fcreated) + ` ` + time(fcreated)", displays the date-time the file was created

The Caption property supports the following keywords:

- **fcount** keyword, returns the number of thumbnail files
- **findex** keyword, specifies the index of the thumbnail file
- **width** keyword, specifies the current width of the thumbnail view
- **height** keyword, specifies the current height of the thumbnail view
- **ffile** keyword, returns the file
- **fname** keyword, returns the name of the file
- **faname** keyword, returns the alternate name of the file, 8.3 format truncated version of the name
- **fext** keyword, returns the extension of the file
- **fcreated** keyword, specifies when a file or directory was created (UTC)
- **fcreated_local** keyword, specifies when a file or directory was created (local date-time)
- **fmodified** keyword, specifies when a file or directory was last modified (UTC)
- **fmodified_local** keyword, specifies when a file or directory was last modified (local date-time)
- **fopened** keyword, specifies when a file or directory was last accessed (UTC)
- **fopened_local** keyword, specifies when a file or directory was last accessed (local date-time)
- **fsize** keyword, specifies the size of the file
- **fsizeF** keyword, returns automatically the size of the file in KB, MB or GB
- **ffolder** keyword, returns 0, 1, 2, -1 or -2, if the object does not exist, it is a file, a shell item, a folder or a drive
- **fattr** keyword, returns the attributes of the file/folder
- **ftype** keyword, returns the type of the file/folder, like JPG File, ...
- **fpicture** keyword, indicates whether the file points to a known picture type, like BMP, PNG, and so on
- **fwidth** keyword, returns the width of the picture file
- **fheight** keyword, returns the height of the picture file
- **ffilter** keyword, returns the currently filter pattern (FilterBarPromptPattern property)
- **fvcount** keyword, indicates the number of visible thumbnails. For instance, if a filter is applied the fvcount returns the number of available/visible/filtered thumbnails.
- **fvindex** keyword, indicates the index of the thumbnail in the visible collection.

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

The Caption property supports the following HTML tags:

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

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

- **<img>key[:width]</img>** inserts a custom size picture into the text being previously loaded using the HTMLPicture property. The Key parameter indicates the key of the picture being displayed. The Width parameter indicates a custom size, if you require to stretch the picture, else the original size of the picture is used.
- **&** glyph characters as **&** (&), **<** (<), **>** (>), **&qout;** (") and **&#number;** (the character with specified code), For instance, the € displays the EUR character. The **&** ampersand is only recognized as markup when it is followed by a known letter or a #character and a digit. For instance if you want to display bold in HTML caption you can use bold
- **<off offset> ... </off>** defines the vertical offset to display the text/element. The offset parameter defines the offset to display the element. This tag is inheritable, so the offset is keep while the associated </off> tag is found. You can use the <off offset> HTML tag in combination with the to define a smaller or a larger font to be displayed. For instance: "Text with <**off** 6>subscript" displays the text such as: Text with $_{\text{subscript}}$ The "Text with <**off** -6>superscript" displays the text such as: Text with ^{subscript}
- **<gra rrggbb;mode;blend> ... </gra>** defines a gradient text. The text color or <fgcolor> defines the starting gradient color, while the rr/gg/bb represents the red/green/blue values of the ending color, 808080 if missing as gray. The mode is a value between 0 and 4, 1 if missing, and blend could be 0 or 1, 0 if missing. The HTML tag can be used to define the height of the font. Any of the rrggbb, mode or blend field may not be specified. The <gra> with no fields, shows a vertical gradient color from the current text color to gray (808080). For instance the "<**gra** FFFFFF;1;1>gradient-center</**gra**>" generates the following picture:

gradient-center

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

outlined

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

shadow

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

outline anti-aliasing

property Thumbnail.CaptionRotate as HTMLRotateEnum

Rotates the HTML caption.

Type	Description
HTMLRotateEnum	A HTMLRotateEnum expression that specifies how the HTML caption is displayed.

By default, the CaptionRotate property is exHTMLHorizontal, which specifies that the HTML text is horizontally displayed. Use the Caption property to specify a caption to be displayed on the thumbnail. The SingleCaption property indicates the expression to generate the HTML caption to be displayed on the thumbnail, when the control shows a single thumbnail (for instance you filter and you got a single result). Use the Alignment property to specify the caption's alignment. Use the Boders / Margins / Padding property to specify the size of the margins where the thumbnail is displayed.

property Thumbnail.Debug as String

Displays debug information.

Type	Description
String	A String expression that specifies the debug information being displayed.

Currently, this property is only for internal use.

property Thumbnail.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. Use the Enabled property to disable the control.

method Thumbnail.EndUpdate ()

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

Type	Description

The BeginUpdate method prevents the control from painting until the EndUpdate method is called. Once that BeginUpdate method was called, you have to make sure that EndUpdate method will be called too.

property Thumbnail.EventParam(Parameter as Long) as Variant

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

Type	Description
Parameter as Long	A long expression that indicates the index of the parameter being requested ie 0 means the first parameter, 1 means the second, and so on. If -1 is used the EventParam property retrieves the number of parameters. If -2, the EventParam gives full information about the event, such as name, identifier, and parameters. Accessing an not-existing parameter produces an OLE error, such as invalid pointer (E_POINTER)
Variant	A VARIANT expression that specifies the parameter's value.

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

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

In most languages you will type something like:

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

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

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

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

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

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

Executes a context menu command.

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

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

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

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

property Thumbnail.ExecuteContextMenu as Long

Executes a command from the object's context menu.

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

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

method Thumbnail.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 Thumbnail1.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 script is composed by lines of instructions. Instructions are separated by "\n\r" (newline) characters.

An instruction can be one of the following:

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

The Template supports the following general functions:

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

property Thumbnail.ExtractFlag as Long

Specifies how the thumbnail is extracted.

Type	Description
Long	A long expression that specifies the flags how the image is to be handled.

The Flag can be one of the following:

- 4 - Used to ask the object to use the supplied aspect ratio
- 32 - Used to tell the object to render as if for the screen
- 512 - higher quality image is requested. For example, if the file is 2000x2000 pixels but the embedded thumbnail is only 100x100 pixels and the user does not set this flag, yet requests a 1000x1000 pixel thumbnail, it always returns the 100x100 pixel thumbnail.

property Thumbnail.ExtractMethod as String

Specifies the order and the methods the control uses to extract the thumbnails.

Type	Description
String	A String expression that specifies the list of methods the control uses to extract the thumbnails, separated by comma. If a method fails, the next one is trying to generate the thumbnail. The first one that succeeds the one is displayed.

By default, the ExtractMethod property is "ThumbnailCache,ThumbnailProvider,ExtractImage". The ExtractMedthod specifies the order and the methods the control uses to extract the thumbnails.

The control is able to extract the thumbnail views using the following system interfaces:

- IPictureDisp
- IExtractIcon
- IExtractImage
- IThumbnailProvider (Windows Vista, Windows 7)
- IThumbnailCache/ISharedBitmap (Windows Vista, Windows 7)

property Thumbnail.ExtractTime as Long

Specifies the time elapsed to extract from last object.

Type	Description
Long	A long expression that specifies the time to extract the thumbnail view

The ExtractTime property retrieves the number of mili-seconds required to extract the thumbnail view from the original object.

property Thumbnail.FilterBarBackColor as Color

Specifies the background color of the control's filter bar.

Type	Description
Color	A color expression that defines the background color for description of the control's filter.

Use the FilterBarForeColor and FilterBarBackColor properties to define the colors used to paint the description for control's filter. Use the BackColor property to specify the control's background color.

property Thumbnail.FilterBarFor as String

Specifies the expression that determines whether the thumbnail is included in the filter.

Type	Description
String	A String expression that defines the expression that determines whether the thumbnail is included in the filter.

By default, the FilterBarFor property is empty, which indicates that the thumbnail's caption is what the filter feature is filtering for. You can use the FilterBarFor property to specify data the filter feature is filtering for. The FilterBarVisible property specifies whether the control's filter bar is visible or hidden. Use the Caption property to specify a caption to be displayed on the thumbnail. The SingleCaption property indicates the expression to generate the HTML caption to be displayed on the thumbnail, when the control shows a single thumbnail.

For instance:

- "fname" filters for file names

The FilterBarFor property supports the following keywords:

- **fcount** keyword, returns the number of thumbnail files
- **findex** keyword, specifies the index of the thumbnail file
- **ffile** keyword, returns the file
- **fname** keyword, returns the name of the file
- **faname** keyword, returns the alternate name of the file, 8.3 format truncated version of the name
- **fext** keyword, returns the extension of the file
- **fcreated** keyword, specifies when a file or directory was created (UTC)
- **fcreated_local** keyword, specifies when a file or directory was created (local date-time)
- **fmodified** keyword, specifies when a file or directory was last modified (UTC)
- **fmodified_local** keyword, specifies when a file or directory was last modified (local date-time)
- **fopened** keyword, specifies when a file or directory was last accessed (UTC)
- **fopened_local** keyword, specifies when a file or directory was last accessed (local date-time)
- **fsize** keyword, specifies the size of the file
- **fsizeF** keyword, returns automatically the size of the file in KB, MB or GB
- **ffolder** keyword, returns 0, 1, 2, -1 or -2, if the object does not exist, it is a file, a shell item, a folder or a drive
- **fattr** keyword, returns the attributes of the file/folder
- **ftype** keyword, returns the type of the file/folder, like JPG File, ...

- **fpicture** keyword, indicates whether the file points to a known picture type, like BMP, PNG, and so on
- **fwidth** keyword, returns the width of the picture file
- **fheight** keyword, returns the height of the picture file
- **ffilter** keyword, returns the currently filter pattern (FilterBarPromptPattern property)
- **fvcount** keyword, indicates the number of visible thumbnails. For instance, if a filter is applied the fvcount returns the number of available/visible/filtered thumbnails.
- **fvindex** keyword, indicates the index of the thumbnail in the visible collection.

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

property Thumbnail.FilterBarForeColor as Color

Specifies the foreground color of the control's filter bar.

Type	Description
Color	A color expression that defines the foreground color of the description of the control's filter.

Use the FilterBarForeColor and FilterBarBackColor properties to define colors used to paint the description of the control's filter.

property Thumbnail.FilterBarPrompt as String

Specifies the caption to be displayed when the filter pattern is missing.

Type	Description
String	A string expression that indicates the HTML caption being displayed in the filter bar, when filter prompt pattern is missing. The FilterBarPromptPattern property specifies the pattern to filter the list using the filter prompt feature.

By default, the FilterBarPrompt property is "<fgcolor=808080>Start Filter...</fgcolor> ". The FilterBarPromptPattern property specifies the pattern to filter the list using the filter prompt feature. Changing the FilterBarPrompt property won't change the current filter. The FilterBarBackColor property specifies the background color or the visual aspect of the control's filter bar. The FilterBarForeColor property specifies the foreground color or the control's filter bar.

The FilterBarPrompt property supports HTML format as described here:

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

rr/gg/bb represents the red/green/blue values of the color in hexa values.

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

FFFFFF;1;1>gradient-center</**gra**>" generates the following picture:

gradient-center

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

outlined

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

shadow

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

outline anti-aliasing

property Thumbnail.FilterBarPromptPattern as String

Specifies the pattern for the filter prompt.

Type	Description
String	A string expression that specifies the pattern to filter the list.

By default, the FilterBarPromptPattern property is empty. The FilterBarPromptPattern property indicates the patter to filter the list. The pattern may include wild characters if the FilterBarPromptType property is exFilterPromptPattern.

property Thumbnail.FilterBarPromptType as FilterPromptEnum

Specifies the type of the filter prompt.

Type	Description
FilterPromptEnum	A FilterPromptEnum expression that specifies how the items are being filtered.

By default, the FilterBarPromptType property is exFilterPromptContainsAll. The filter prompt feature allows you to filter the items as you type while the filter bar is visible on the bottom part of the list area. The Filter prompt feature allows at runtime filtering data on hidden columns too. Use the FilterBarPrompt property to specify the HTML caption being displayed in the filter bar when the filter pattern is missing. The FilterBarPromptPattern property specifies the pattern to filter the list.

The FilterBarPromptType property supports the following values:

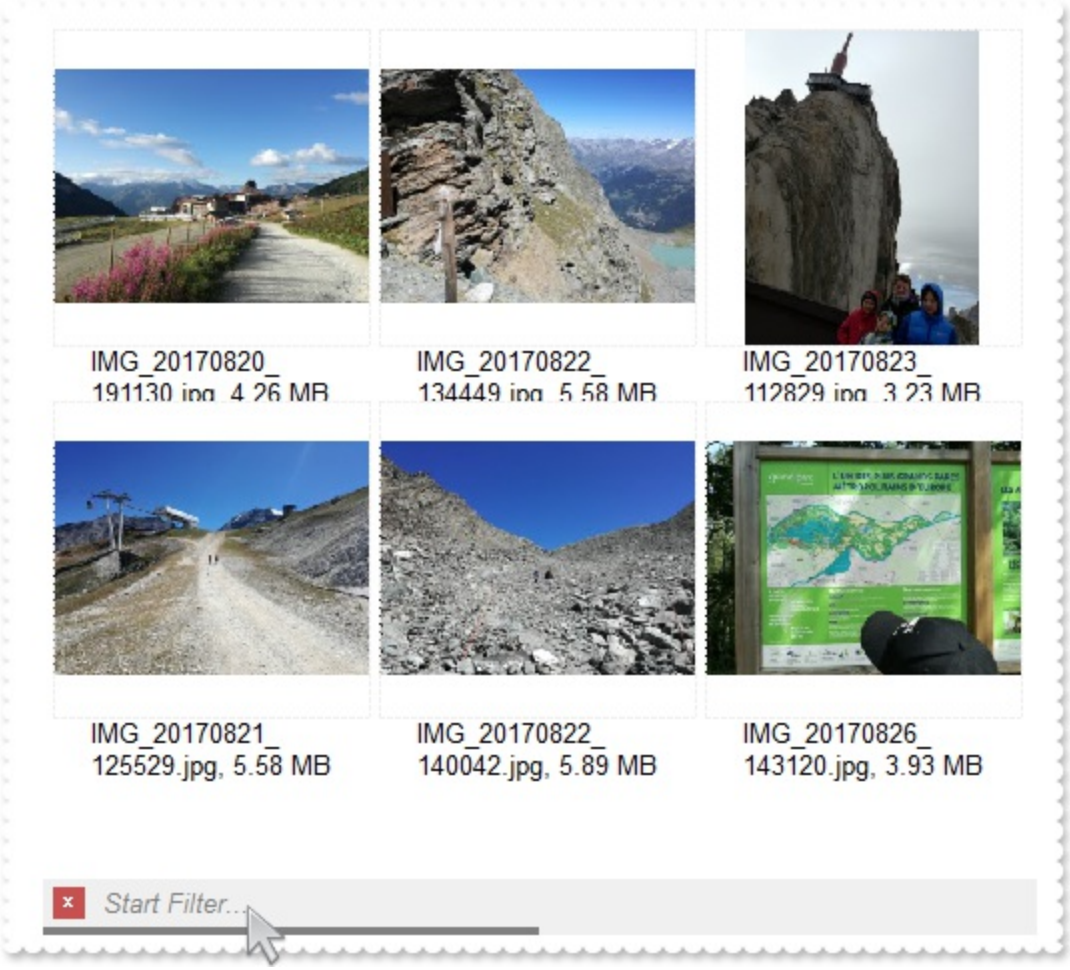
- **exFilterPromptContainsAll**, The list includes the items that contains all specified sequences in the filter (FilterBarPromptPattern property). Can be combined with exFilterPromptCaseSensitive, exFilterPromptStartWords, exFilterPromptEndWords or exFilterPromptWords
- **exFilterPromptContainsAny**, The list includes the items that contains any of specified sequences in the filter (FilterBarPromptPattern property). Can be combined with exFilterPromptCaseSensitive, exFilterPromptStartWords, exFilterPromptEndWords or exFilterPromptWords
- **exFilterPromptStartWith**, The list includes the items that starts with any specified sequences in the filter (FilterBarPromptPattern property). Can be combined with exFilterPromptCaseSensitive, exFilterPromptStartWords, exFilterPromptEndWords or exFilterPromptWords
- **exFilterPromptEndWith**, The list includes the items that ends with any specified sequences in the filter (FilterBarPromptPattern property). Can be combined with exFilterPromptCaseSensitive, exFilterPromptStartWords, exFilterPromptEndWords or exFilterPromptWords
- **exFilterPromptPattern**, The filter indicates a pattern that may include wild characters to be used to filter the items in the list. The FilterBarPromptPattern property may include wild characters as follows:
  - '?' for any single character
  - '*' for zero or more occurrences of any character
  - '#' for any digit character
  - ' ' space delimits the patterns inside the filter

property Thumbnail.FilterBarVisible as FilterBarVisibleEnum

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

Type	Description
FilterBarVisibleEnum	A FilterBarVisibleEnum expression that defines the way the control's filter bar is shown.

By default, The FilterBarPromptVisible property is exFilterBarVisible. The filter prompt feature allows you to filter the files as you type while the filter bar is visible on the bottom part of the list area. The FilterBarFor property specifies the expression that determines whether the thumbnail is included in the filter. Use the FilterBarPrompt property to specify the HTML caption being displayed in the filter bar when the filter pattern is missing. The FilterBarPromptPattern property specifies the pattern to filter the list. The FilterBarPromptType property specifies the type of filtering when the user edits the prompt in the filter bar.



property Thumbnail.Font as IFontDisp

Retrieves or sets the control's font.

Type	Description
IFontDisp	A Font object to paint the control's caption.

Use the Font property to change the font to display the control's caption. Use the Caption property to display a caption on the thumbnail control. The SingleCaption property indicates the expression to generate the HTML caption to be displayed on the thumbnail, when the control shows a single thumbnail (for instance you filter and you got a single result). Use the ForeColor property to specify the control's foreground color. Also, you can use the <font> HTML tag to display a portion of text using a different font.

property Thumbnail.ForeColor as Color

Specifies the control's foreground color.

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

Use the ForeColor property to specify the caption's foreground color. Use the Caption property to display a caption on the thumbnail control. Use the Font property to specify the font to display the caption on the thumbnail. Use the Transparency property to specify the transparency to display the caption on the thumbnail control. Use the InputFile property to specify the file to display its thumbnail. Use the OutputFile property to save the thumbnail picture. You can use the <fgcolor> HTML tag to draw portion of text using different foreground color.

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

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

Type	Description
Expression as String	A String that defines the expression to be evaluated.
A as Variant	A VARIANT expression that indicates the value of the A keyword.
B as Variant	A VARIANT expression that indicates the value of the B keyword.
C as Variant	A VARIANT expression that indicates the value of the C keyword.
File as Variant	A string expression that indicates a file to be queried for properties like fname, fext and so on.

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

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

For instance:

- "A + B + C", adds / concatenates the values of the A, B and C
- "value MIN 0 MAX 99", limits the value between 0 and 99
- "value format ``", formats the value with two decimals, according to the control's panel setting
- "date(`now`)" returns the current time as double
- FormatABC("ffile +` ` + (len(fname) ? `this is a file/folder`: `not found`)",,,,"C:\Program Files\Exontrol\ExThumbnail\Sample\elogo.jpg"), determines wgether rge specified file exists

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

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

If the File parameter is not empty, and refer a valid file, the FormatABC method support the

following keywords:

- **ffile** keyword, returns the file
- **fname** keyword, returns the name of the file
- **faname** keyword, returns the alternate name of the file, 8.3 format truncated version of the name
- **fext** keyword, returns the extension of the file
- **fcreated** keyword, specifies when a file or directory was created (UTC)
- **fcreated_local** keyword, specifies when a file or directory was created (local date-time)
- **fmodified** keyword, specifies when a file or directory was last modified (UTC)
- **fmodified_local** keyword, specifies when a file or directory was last modified (local date-time)
- **fopened** keyword, specifies when a file or directory was last accessed (UTC)
- **fopened_local** keyword, specifies when a file or directory was last accessed (local date-time)
- **fsize** keyword, specifies the size of the file
- **fsizeF** keyword, returns automatically the size of the file in KB, MB or GB
- **ffolder** keyword, returns 0, 1, 2, -1 or -2, if the object does not exist, it is a file, a shell item, a folder or a drive
- **fattr** keyword, returns the attributes of the file/folder
- **ftype** keyword, returns the type of the file/folder, like JPG File, ...
- **fpicture** keyword, indicates whether the file points to a known picture type, like BMP, PNG, and so on
- **fwidth** keyword, returns the width of the picture file
- **fheight** keyword, returns the height of the picture file
- **ffilter** keyword, returns the currently filter pattern (FilterBarPromptPattern property)
- **fvcount** keyword, indicates the number of visible thumbnails. For instance, if a filter is applied the fvcount returns the number of available/visible/filtered thumbnails.
- **fvindex** keyword, indicates the index of the thumbnail in the visible collection.

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

property Thumbnail.FormatAnchor(New as Boolean) as String

Specifies the visual effect for anchor elements in HTML captions.

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

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

method Thumbnail.FreezeEvents (Freeze as Boolean)

Prevents the control to fire any event.

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

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

property Thumbnail.HTMLPicture(Key as String) as Variant

Adds or replaces a picture in HTML captions.

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

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

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

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

<CONTROL>.Caption = "A <img>pic1</img>"
```

property Thumbnail.hWnd as Long

Retrieves the control's window handle.

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

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

method Thumbnail.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 user can add images at design time, by drag and drop files to combo's image holder. The ImageSize property defines the size (width/height) of the icons within the control's Images collection. Use the RepaceIcon method to add, remove or clear icons in the control's images collection. Use the <img> HTML tag to display icons in the control's caption.

property Thumbnail.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 Thumbnail.InputFile as String

Specifies the file to display the thumbnail.

Type	Description
String	A String expression that specifies the file to display its thumbnail or a web page to get its thumbnail (if it starts with http://).

By default, the InputFile property is empty. The InputFile property removes any previously files. Use the InputFile property to programmatically display a thumbnail for a specified file. Use the OutputFile property to save the thumbnail picture. Use the Borders property to specify the size of the margins where the thumbnail is displayed. Use the Thumbnail property to retrieve the thumbnail picture. Use the ThumbnailType property to specify whether the thumbnail is a bitmap or an icon. For instance, if you set the ThumbnailType property on exThumbnailIcon, and calling the InputFile property, the control gets the icon representation of the file. If the ThumbnailType property on exThumbnailIcon is exThumbnailBitmap the control tries to get a bitmap representation of the file if it is available, else the Thumbnail property is nothing, and the ThumbnailType property is exNoThumbnail. The control fires the Changing event before updating the thumbnail view, and the Change event once the thumbnail is changed. You can clear the files into the control by calling the InputFile property on "". Use the AcceptFiles property to disable drag-and-drop files in the thumbnail control. The AcceptFolders property specifies whether the control accepts drag-and-drop folders. The LimitInputFiles property limits the number of files the control can display.

The control is able to extract the thumbnail views using the following system interfaces:

- IPictureDisp
- IExtractIcon
- IExtractImage
- IThumbnailProvider (Windows Vista, Windows 7)
- IThumbnailCache/ISharedBitmap (Windows Vista, Windows 7)

The ExtractMedthod specifies the order and the methods the control uses to extract the thumbnails.

You can use any of the following properties/method to add files into the control:

- InputFile property specifies the file or the web page to display its thumbnail.
- InputFiles property indicates the list of files to be thumbnailed
- AddInputFiles method adds files to be thumbnailed.

property Thumbnail.InputFiles as Variant

Indicates the list of files to be thumbnailed.

Type	Description
Variant	A string expression that specifies a list of files to be added (separated by \r\n sequence) or a safe array of variant, where each element could be a list of files to be added (separated by \r\n sequence)

The InputFiles method removes any previously files, and adds specified files to the control. You can clear the files into the control by calling the InputFile property on "". The control fires the Changing event before updating the thumbnail view, and the Change event once the thumbnail is changed. Use the ThumbnailType property to specify whether the thumbnail is a bitmap or an icon. Use the AcceptFiles property to disable drag-and-drop files in the thumbnail control. The AcceptFolders property specifies whether the control accepts drag-and-drop folders. The LimitInputFiles property limits the number of files the control can display.

You can use any of the following properties/method to add files into the control:

- InputFile property specifies the file or the web page to display its thumbnail.
- InputFiles property indicates the list of files to be thumbnailed
- AddInputFiles method adds files to be thumbnailed.

property Thumbnail.KeepOriginalThumbnail as Boolean

Specifies whether the thumbnail is shown as it is provided.

Type	Description
Boolean	A boolean expression that specifies whether the thumbnail is displayed as it is provided, in other words without using a transparent backcolor.

By default, the KeepOriginalThumbnail property is False. Use the KeepOriginalThumbnail property to retrieve bitmap thumbnails using transparent background color (implementation removes the black part of the thumbnail view). Use the BackColor property to change the control's background color. Use the Borders property to specify the size of the margins where the thumbnail picture is displayed. Use the Picture property to display a picture on the control's background. Use the PictureDisplay property to layout the control's Picture on the background.

property Thumbnail.LimitInputFiles as Long

Limits the number of files the control can display.

Type	Description
Long	A long expression that specifies the number of files the control can display.

By default, the LimitInputFiles property is 0, which indicates no limit . The LimitInputFiles property limits the number of files the control can display.

You can use any of the following properties/method to add files into the control:

- InputFile property specifies the file or the web page to display its thumbnail.
- InputFiles property indicates the list of files to be thumbnailed
- AddInputFiles method adds files to be thumbnailed.

property Thumbnail.Margins as String

Specifies the distance between thumbnails, when multiple thumbnails are displayed on the control.

Type	Description
String	A String expression that specifies the distance between thumbnails, when multiple thumbnails are displayed on the control.

By default, the Margins property is "4 4". Use the Alignment property to specify the caption being displayed on the thumbnail control. Use the Transparency property to specify the percent of transparency being used to paint the caption on the thumbnail. Use the Appearance property to specify whether the thumbnail displays a border.

You can use the following properties to define the distance and padding of the thumbnails

- The Borders property indicates the margins where the thumbnail is displayed, relative to the control's border.
- The Margins property specifies the distance between thumbnails, when multiple thumbnails are displayed on the control.
- The Padding property generates space around thumbnail. Clears the area around the content (inside the border) of the thumbnail.

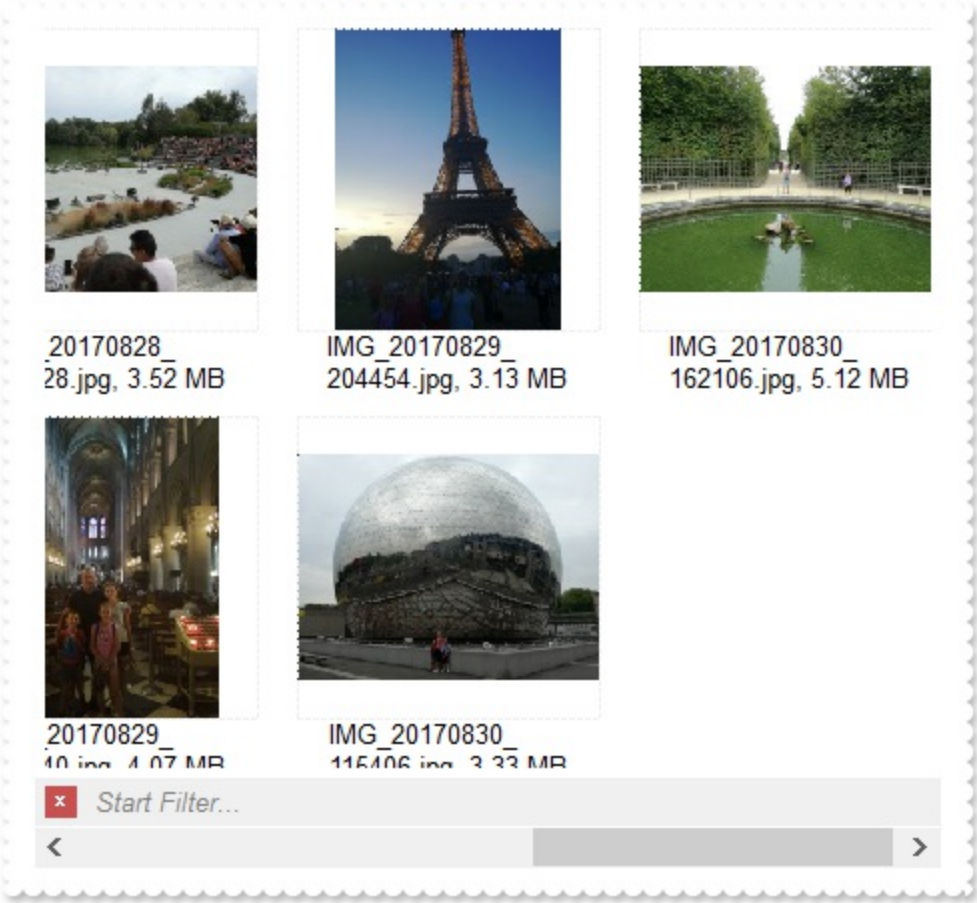
property Thumbnail.Mode as ThumbnailModeEnum

Specifies how thumbnails are arranged on the control's client area.

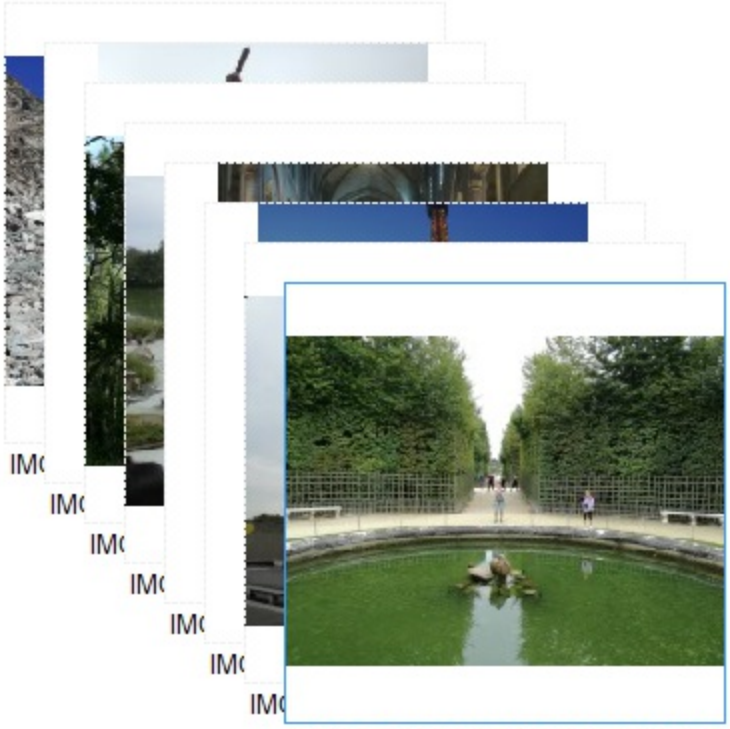
Type	Description
ThumbnailModeEnum	A ThumbnailModeEnum expression that specifies the mode the thumbnails are arranged on the control.

By default, the Mode property is exThumbnailGrid | exThumbnailAutoFit | exThumbnailStretch | exThumbnailCenter | exThumbnailAllowResize | exThumbnailAutoFitOnDblClk | exThumbnailKeepAspectRatio. The Mode property specifies how thumbnails are arranged on the control's client area. The Sort method sorts files being displayed in the control.

The following screen shot shows the exThumbnailGrid mode:



The following screen shot shows the exThumbnailStack mode:



IMG_20170830_162106.jpg, 5.12 MB

Start Filter...

method Thumbnail.OLEDrag ()

Causes a component to initiate an OLE drag/drop operation.

Type	Description

Only for internal use.

property Thumbnail.OLEDropMode as exOLEDropModeEnum

Returns or sets how a target component handles drop operations

Type	Description
exOLEDropModeEnum	An exOLEDropModeEnum expression that indicates the OLE Drag and Drop mode.

In the /NET Assembly, you have to use the AllowDrop property as explained here:

- *https://www.exontrol.com/sg.jsp?content=support/faq/net/#dragdrop*

In the /COM version, you have to set the OLEDropMode property as explained here:

- *https://www.exontrol.com/sg.jsp?content=support/faq/all/#dragdrop*

By default, the OLEDropMode property is exOLEDropNone. Currently, the ExThumbnail control supports only manual OLE Drag and Drop operation. See the OLEStartDrag and OLEDragDrop events for more details about implementing drag and drop operations into the ExThumbnail control.

method Thumbnail.OutputFile (Output as String)

Specifies the file where to save the thumbnail.

Type	Description
Output as String	A String expression that specifies the path and the filename where the thumbnail picture is saved.

Call the OutputFile property to programmatically save the thumbnail picture as BMP or ICO. The OutputFile method fails if there is no thumbnail loaded. Use the InputFile property to specify a file whose thumbnail should be get. The .bmp suffix is appended to the output if the thumbnail is a bitmap, else .ico is added. Use the ThumbnailType property to retrieve the type of the thumbnail being retrieved. Use the Thumbnail property to access the thumbnail as a picture. The SaveAs method generates programmatically pictures from the current thumbnail, as BMP, JPG, GIF, PNG, TIFF format.

property Thumbnail.Padding as String

Generates space around thumbnail. Clears the area around the content (inside the border) of the thumbnail.

Type	Description
String	A String expression that specifies the padding to left, top, right and bottom margins

By default, the Borders property is "0 0 0 18". Use the Alignment property to specify the caption being displayed on the thumbnail control. Use the Transparency property to specify the percent of transparency being used to paint the caption on the thumbnail. Use the Appearance property to specify whether the thumbnail displays a border.

You can use the following properties to define the distance and padding of the thumbnails

- The Borders property indicates the margins where the thumbnail is displayed, relative to the control's border.
- The Margins property specifies the distance between thumbnails, when multiple thumbnails are displayed on the control.
- The Padding property generates space around thumbnail. Clears the area around the content (inside the border) of the thumbnail.

property Thumbnail.Picture as IPictureDisp

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

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

By default, the control has no picture associated. The control uses the PictureDisplay property to determine how the picture is displayed on the control's background. Use the InputFile property to display the file's thumbnail. Use the Thumbnail property to retrieve the thumbnail picture. Use the ThumbnailType property to specify whether the thumbnail is a bitmap or an icon. Use the BackColor property to specify the control's background color.

property Thumbnail.PictureDisplay as PictureDisplayEnum

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

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

By default, the PictureDisplay property is exTile. Use the PictureDisplay property specifies how the Picture is displayed on the control's background. If the control has no picture associated the PictureDisplay property has no effect. Use the InputFile property to display the file's thumbnail. Use the Thumbnail property to retrieve the thumbnail picture. Use the ThumbnailType property to specify whether the thumbnail is a bitmap or an icon. Use the BackColor property to specify the control's background color.

method Thumbnail.Refresh ()

Refreshes the control.

Type	Description

The Refresh method forces repainting the control. Use the BeginUpdate and EndUpdate methods to maintain the control's performance while adding multiple items or columns. Use the hWnd property to get the handle of the control's window.

The following VB sample calls the Refresh method:

```
Thumbnail1.Refresh
```

The following C++ sample calls the Refresh method:

```
m_thumbnail.Refresh();
```

The following VB.NET sample calls the Refresh method:

```
AxThumbnail1.CtlRefresh()
```

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

The following C# sample calls the Refresh method:

```
axThumbnail1.CtlRefresh();
```

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

The following VFP sample calls the Refresh method:

```
thisform.Thumbnail1.Object.Refresh()
```

method Thumbnail.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.
Index as Variant	A long expression that indicates the index where icon is inserted.

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

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

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

i = ExThumbnail1.ReplaceIcon(LoadPicture("d:\icons\help.ico").Handle), i specifies the index where the icon is added

The following VB sample replaces an icon into control's images list::

i = ExThumbnail1.ReplaceIcon(LoadPicture("d:\icons\help.ico").Handle, 0), i is zero, so the first icon is replaced.

The following VB sample removes an icon from control's images list:

ExThumbnail1.ReplaceIcon 0, i, i specifies the index of icon removed.

The following VB clears the control's icons collection:

ExThumbnail1.ReplaceIcon 0, -1

method Thumbnail.SaveAs (File as String, [Width as Variant], [Height as Variant])

Save the thumbnail, as a picture file in specified format giving by the extension (characters after last dot, determines the graphical/ format of the file).

Type	Description
File as String	A String expression that specifies the picture-file to be generated. The extension of the file (characters after last dot) determines the graphical/ format of the file to be saved as follows: • ***.bmp *.dib *.rle**, exports the pages to **BMP** format. • ***.jpg *.jpe *.jpeg *.jfif**, exports the pages to **JPEG** format. • ***.gif**, , exports the pages to **GIF** format. • ***.tif *.tiff**, exports the pages to **TIFF** format. • ***.png**, exports the pages to **PNG** format.
Width as Variant	A Long expression that specifies the width of the picture to be generated. If missing, it generates a picture of 120-pixels wide.
Height as Variant	A Long expression that specifies the height of the picture to be generated. If missing, it generates a picture of 120-pixels tall.

The SaveAs method generates programmatically pictures from the current thumbnail, as BMP, JPG, GIF, PNG, TIFF format. Use the InputFile property to specify a file whose thumbnail should be get. Call the OutputFile property to programmatically save the thumbnail picture as BMP or ICO. Use the Thumbnail property to access the thumbnail as a picture. Use the ThumbnailType property to retrieve the type of the thumbnail being retrieved. For instance, you can use the SaveAs method to generates a BMP file from an EBN object, or from a PDF or CAD file.

property Thumbnail.ScrollButtonHeight as Long

Specifies the height of the button in the vertical scrollbar.

Type	Description
Long	A long expression that defines the height of the button in the vertical scroll bar.

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

property Thumbnail.ScrollButtonWidth as Long

Specifies the width of the button in the horizontal scrollbar.

Type	Description
Long	A long expression that defines the width of the button in the horizontal scroll bar.

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

property Thumbnail.ScrollFont (ScrollBar as ScrollBarEnum) as IFontDisp

Retrieves or sets the scrollbar's font.

Type	Description
ScrollBar as ScrollBarEnum	A ScrollBarEnum expression that indicates the vertical or the horizontal scroll bar.
IFontDisp	A Font object

Use the ScrollFont property to specify the font in the control's scroll bar. Use the ScrolPartCaption property to specify the caption of the scroll's part. Use the ScrollPartVisible property to add or remove buttons/parts in the control's scrollbar. By default, when a part becomes visible, the ScrollPartEnable property is automatically called, so the parts becomes enabled. Use the ScrollPartEnable property to specify enable or disable parts in the control's scrollbar.

property Thumbnail.ScrollHeight as Long

Specifies the height of the horizontal scrollbar.

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

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

property Thumbnail.ScrollOrderParts(ScrollBar as ScrollBarEnum) as String

Specifies the order of the buttons in the scroll bar.

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

Use the ScrollOrderParts to customize the order of the buttons in the scroll bar. By default, the ScrollOrderParts property is empty. If the ScrollOrderParts property is empty the default order of the buttons in the scroll bar are displayed like follows:

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

Use the ScrollOrderParts property to change the order of the buttons in the scroll bar. For instance, "l,r,t,l1,r1" puts the left and right buttons to the left of the thumb area, and the l1 and r1 buttons right after the thumb area. If the parts are not specified in the ScrollOrderParts property, automatically they are added to the end.

The list of supported literals in the ScrollOrderParts property is:

- **l** for exLeftBPart, (<) The left or top button.
- **l1** for exLeftB1Part, (L1) The first additional button, in the left or top area.
- **l2** for exLeftB2Part, (L2) The second additional button, in the left or top area.
- **l3** for exLeftB3Part, (L3) The third additional button, in the left or top area.
- **l4** for exLeftB4Part, (L4) The forth additional button, in the left or top area.
- **l5** for exLeftB5Part, (L5) The fifth additional button, in the left or top area.
- **t** for exLowerBackPart, exThumbPart and exUpperBackPart, The union between the exLowerBackPart and the exUpperBackPart parts.
- **r** for exRightBPart, (>) The right or down button.

- **r1** for exRightB1Part, (R1) The first additional button in the right or down side.
- **r2** for exRightB2Part, (R2) The second additional button in the right or down side.
- **r3** for exRightB3Part, (R3) The third additional button in the right or down side.
- **r4** for exRightB4Part, (R4) The forth additional button in the right or down side.
- **r5** for exRightB5Part, (R5) The fifth additional button in the right or down side.
- **r6** for exRightB6Part, (R6) The sixth additional button in the right or down side.

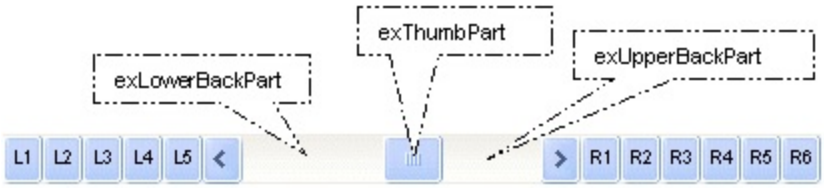
Any other literal between commas is ignored. If duplicate literals are found, the second is ignored, and so on. For instance, "t,l,r" indicates that the left/top and right/bottom buttons are displayed right/bottom after the thumb area.

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

Specifies the caption being displayed on the specified scroll part.

Type	Description
ScrollBar as ScrollBarEnum	A ScrollBar expression that indicates the scrollbar where the caption is displayed.
Part as ScrollPartEnum	A ScrollPartEnum expression that specifies the parts of the scroll where the text is displayed
String	A String expression that specifies the caption being displayed on the part of the scroll bar.

Use the ScrolPartCaption property to specify the caption of the scroll's part. Use the ScrollPartVisible property to add or remove buttons/parts in the control's scrollbar. By default, when a part becomes visible, the ScrollPartEnable property is automatically called, so the parts becomes enabled. Use the ScrollPartEnable property to specify enable or disable parts in the control's scrollbar. Use the ScrollFont property to specify the font in the control's scroll bar. Use the ScrollOrderParts property to customize the order of the buttons in the scroll bar. The ScrollPartCaptionAlignment property specifies the alignment of the caption in the part of the scroll bar.



By default, the following parts are shown:

- exLeftBPart (the left or up button of the control)
- exLowerBackPart (the part between the left/up button and the thumb part of the control)
- exThumbPart (the thumb/scrollbox part)
- exUpperBackPart (the part between the the thumb and the right/down button of the control)
- exRightBPart (the right or down button of the control)

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

```
With Thumbnail1
  .BeginUpdate
```

```
        .ScrollBars = exDisableBoth
        .ScrollPartVisible(exVScroll, exLeftB1Part Or exRightB1Part) = True
        .ScrollPartCaption(exVScroll, exLeftB1Part) = "<img></img>1"
        .ScrollPartCaption(exVScroll, exRightB1Part) = "<img></img>2"
    .EndUpdate
End With
```

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

```
With AxThumbnail1
    .BeginUpdate()
    .ScrollBars = EXTHUMBNAILLib.ScrollBarsEnum.exDisableBoth
    .set_ScrollPartVisible(EXTHUMBNAILLib.ScrollBarEnum.exVScroll,
EXTHUMBNAILLib.ScrollPartEnum.exLeftB1Part Or
EXTHUMBNAILLib.ScrollPartEnum.exRightB1Part, True)
    .set_ScrollPartCaption(EXTHUMBNAILLib.ScrollBarEnum.exVScroll,
EXTHUMBNAILLib.ScrollPartEnum.exLeftB1Part, "<img></img>1")
    .set_ScrollPartCaption(EXTHUMBNAILLib.ScrollBarEnum.exVScroll,
EXTHUMBNAILLib.ScrollPartEnum.exRightB1Part, "<img></img>2")
    .EndUpdate()
End With
```

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

```
axThumbnail1.BeginUpdate();
axThumbnail1.ScrollBars = EXTHUMBNAILLib.ScrollBarsEnum.exDisableBoth;
axThumbnail1.set_ScrollPartVisible(EXTHUMBNAILLib.ScrollBarEnum.exVScroll,
EXTHUMBNAILLib.ScrollPartEnum.exLeftB1Part |
EXTHUMBNAILLib.ScrollPartEnum.exRightB1Part, true);
axThumbnail1.set_ScrollPartCaption(EXTHUMBNAILLib.ScrollBarEnum.exVScroll,
EXTHUMBNAILLib.ScrollPartEnum.exLeftB1Part , "<img></img>1");
axThumbnail1.set_ScrollPartCaption(EXTHUMBNAILLib.ScrollBarEnum.exVScroll,
EXTHUMBNAILLib.ScrollPartEnum.exRightB1Part, "<img></img>2");
axThumbnail1.EndUpdate();
```

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

```
m_thumbnail.BeginUpdate();
m_thumbnail.SetScrollBars( 15 /*exDisableBoth*/ );
m_thumbnail.SetScrollPartVisible( 0 /*exVScroll*/, 32768  /*exLeftB1Part*/ | 32
/*exRightB1Part*/, TRUE );
m_thumbnail.SetScrollPartCaption( 0 /*exVScroll*/, 32768  /*exLeftB1Part*/ , _T("<img>
</img>1") );
m_thumbnail.SetScrollPartCaption( 0 /*exVScroll*/, 32 /*exRightB1Part*/ , _T("<img>
</img>2") );
m_thumbnail.EndUpdate();
```

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

```
With thisform.Thumbnail1
   .BeginUpdate
      .ScrollBars = 15
      .ScrollPartVisible(0, bitor(32768,32)) = .t.
      .ScrollPartCaption(0,32768) = "<img></img>1"
      .ScrollPartCaption(0, 32) = "<img></img>2"
   .EndUpdate
EndWith
```

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

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

property Thumbnail.ScrollPartCaptionAlignment(ScrollBar as ScrollBarEnum, Part as ScrollPartEnum) as AlignmentEnum

Specifies the alignment of the caption in the part of the scroll bar.

Type	Description
ScrollBar as ScrollBarEnum	A ScrollBar expression that indicates the scrollbar where the caption is displayed.
Part as ScrollPartEnum	A ScrollPartEnum expression that specifies the parts of the scroll where the text is displayed
AlignmentEnum	An AlignmentEnum expression that specifies the alignment of the caption in the part of the scrollbar.

The ScrollPartCaptionAlignment property specifies the alignment of the caption in the part of the scroll bar. By default, the caption is centered. Use the ScrolPartCaption property to specify the caption being displayed on specified part of the scroll bar. Use the ScrollPartVisible property to add or remove buttons/parts in the control's scrollbar.

The following VB sample displays "left" aligned to the left on the lower part of the control's horizontal scroll bar, and "right" aligned to the right on the upper part of the control's horizontal scroll bar:

```
With Thumbnail1
   .ScrollPartCaption(exHScroll,exLowerBackPart) = "left"
   .ScrollPartCaptionAlignment(exHScroll,exLowerBackPart) = LeftAlignment
   .ScrollPartCaption(exHScroll,exUpperBackPart) = "right"
   .ScrollPartCaptionAlignment(exHScroll,exUpperBackPart) = RightAlignment
   .ColumnAutoResize = False
   .Columns.Add 1
   .Columns.Add 2
   .Columns.Add 3
   .Columns.Add 4
End With
```

The following VB.NET sample displays "left" aligned to the left on the lower part of the control's horizontal scroll bar, and "right" aligned to the right on the upper part of the control's horizontal scroll bar:

```
With AxThumbnail1
```

```
.set_ScrollPartCaption(EXTHUMBNAILLib.ScrollBarEnum.exHScroll,EXTHUMBNAILLib.ScrollP

.set_ScrollPartCaptionAlignment(EXTHUMBNAILLib.ScrollBarEnum.exHScroll,EXTHUMBNAIL

.set_ScrollPartCaption(EXTHUMBNAILLib.ScrollBarEnum.exHScroll,EXTHUMBNAILLib.ScrollP

.set_ScrollPartCaptionAlignment(EXTHUMBNAILLib.ScrollBarEnum.exHScroll,EXTHUMBNAIL

    .ColumnAutoResize = False
    .Columns.Add 1
    .Columns.Add 2
    .Columns.Add 3
    .Columns.Add 4
End With
```

The following C# sample displays "left" aligned to the left on the lower part of the control's horizontal scroll bar, and "right" aligned to the right on the upper part of the control's horizontal scroll bar:

```
axThumbnail1.set_ScrollPartCaption(EXTHUMBNAILLib.ScrollBarEnum.exHScroll,EXTHUMBN

axThumbnail1.set_ScrollPartCaptionAlignment(EXTHUMBNAILLib.ScrollBarEnum.exHScroll,E

axThumbnail1.set_ScrollPartCaption(EXTHUMBNAILLib.ScrollBarEnum.exHScroll,EXTHUMBN

axThumbnail1.set_ScrollPartCaptionAlignment(EXTHUMBNAILLib.ScrollBarEnum.exHScroll,E

axThumbnail1.ColumnAutoResize = false;
axThumbnail1.Columns.Add(1.ToString());
axThumbnail1.Columns.Add(2.ToString());
axThumbnail1.Columns.Add(3.ToString());
axThumbnail1.Columns.Add(4.ToString());
```

The following C++ sample displays "left" aligned to the left on the lower part of the control's horizontal scroll bar, and "right" aligned to the right on the upper part of the control's

horizontal scroll bar:

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

   #import "ExThumbnail.dll"
   using namespace EXTHUMBNAILLib;
*/
EXTHUMBNAILLib::IThumbnailPtr spThumbnail1 = GetDlgItem(IDC_THUMBNAIL1)-
>GetControlUnknown();
spThumbnail1-
>PutScrollPartCaption(EXTHUMBNAILLib::exHScroll,EXTHUMBNAILLib::exLowerBackPart,L"l

spThumbnail1-
>PutScrollPartCaptionAlignment(EXTHUMBNAILLib::exHScroll,EXTHUMBNAILLib::exLowerB

spThumbnail1-
>PutScrollPartCaption(EXTHUMBNAILLib::exHScroll,EXTHUMBNAILLib::exUpperBackPart,L"r

spThumbnail1-
>PutScrollPartCaptionAlignment(EXTHUMBNAILLib::exHScroll,EXTHUMBNAILLib::exUpperB

spThumbnail1->PutColumnAutoResize(VARIANT_FALSE);
spThumbnail1->GetColumns()->Add(L"1");
spThumbnail1->GetColumns()->Add(L"2");
spThumbnail1->GetColumns()->Add(L"3");
spThumbnail1->GetColumns()->Add(L"4");
```

The following VFP sample displays "left" aligned to the left on the lower part of the control's horizontal scroll bar, and "right" aligned to the right on the upper part of the control's horizontal scroll bar:

```
with thisform.Thumbnail1
   .ScrollPartCaption(1,512) = "left"
   .ScrollPartCaptionAlignment(1,512) = 0
   .ScrollPartCaption(1,128) = "right"
```

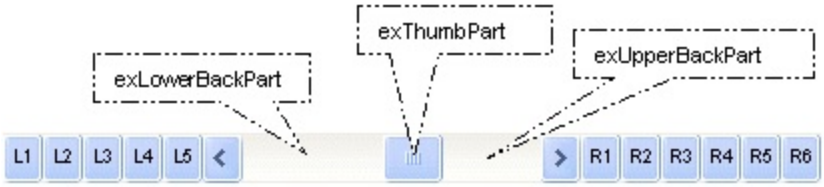
```
   .ScrollPartCaptionAlignment(1,128) = 2
   .ColumnAutoResize = .F.
   .Columns.Add(1)
   .Columns.Add(2)
   .Columns.Add(3)
   .Columns.Add(4)
endwith
```

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

Indicates whether the specified scroll part is enabled or disabled.

Type	Description
ScrollBar as ScrollBarEnum	A ScrollBar expression that indicates the scrollbar where the part is enabled or disabled.
Part as ScrollPartEnum	A ScrollPartEnum expression that specifies the parts of the scroll bar being enabled or disabled.
Boolean	A Boolean expression that specifies whether the scrollbar's part is enabled or disabled.

By default, when a part becomes visible, the ScrollPartEnable property is automatically called, so the parts becomes enabled. Use the ScrollPartVisible property to add or remove buttons/parts in the control's scrollbar. Use the ScrollPartEnable property to specify enable or disable parts in the control's scrollbar. Use the ScrolPartCaption property to specify the caption of the scroll's part. Use the ScrollOrderParts property to customize the order of the buttons in the scroll bar.

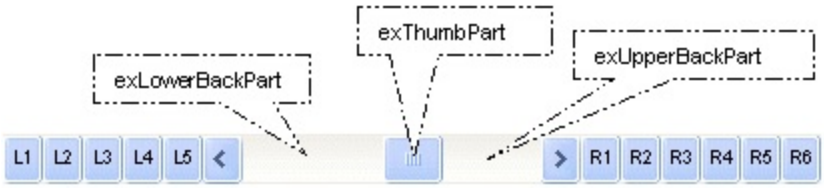


property Thumbnail.ScrollPartVisible(ScrollBar as ScrollBarEnum, Part as ScrollPartEnum) as Boolean

Indicates whether the specified scroll part is visible or hidden.

Type	Description
ScrollBar as ScrollBarEnum	A ScrollBar expression that indicates the scrollbar where the part is visible or hidden.
Part as ScrollPartEnum	A ScrollPartEnum expression that specifies the parts of the scroll bar being visible
Boolean	A Boolean expression that specifies whether the scrollbar's part is visible or hidden.

Use the ScrollPartVisible property to add or remove buttons/parts in the control's scrollbar. By default, when a part becomes visible, the ScrollPartEnable property is automatically called, so the parts becomes enabled. Use the ScrollPartEnable property to specify enable or disable parts in the control's scrollbar. Use the ScrolPartCaption property to specify the caption of the scroll's part. Use the Background property to change the visual appearance for any part in the control's scroll bar. Use the ScrollOrderParts property to customize the order of the buttons in the scroll bar.



By default, the following parts are shown:

- exLeftBPart (the left or up button of the control)
- exLowerBackPart (the part between the left/up button and the thumb part of the control)
- exThumbPart (the thumb/scrollbox part)
- exUpperBackPart (the part between the the thumb and the right/down button of the control)
- exRightBPart (the right or down button of the control)

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

```
With Thumbnail1
  .BeginUpdate
    .ScrollBars = exDisableBoth
```

```
        .ScrollPartVisible(exVScroll, exLeftB1Part Or exRightB1Part) = True
        .ScrollPartCaption(exVScroll, exLeftB1Part) = "<img></img>1"
        .ScrollPartCaption(exVScroll, exRightB1Part) = "<img></img>2"
    .EndUpdate
End With
```

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

```
With AxThumbnail1
    .BeginUpdate()
    .ScrollBars = EXTHUMBNAILLib.ScrollBarsEnum.exDisableBoth
    .set_ScrollPartVisible(EXTHUMBNAILLib.ScrollBarEnum.exVScroll,
EXTHUMBNAILLib.ScrollPartEnum.exLeftB1Part Or
EXTHUMBNAILLib.ScrollPartEnum.exRightB1Part, True)
    .set_ScrollPartCaption(EXTHUMBNAILLib.ScrollBarEnum.exVScroll,
EXTHUMBNAILLib.ScrollPartEnum.exLeftB1Part, "<img></img>1")
    .set_ScrollPartCaption(EXTHUMBNAILLib.ScrollBarEnum.exVScroll,
EXTHUMBNAILLib.ScrollPartEnum.exRightB1Part, "<img></img>2")
    .EndUpdate()
End With
```

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

```
axThumbnail1.BeginUpdate();
axThumbnail1.ScrollBars = EXTHUMBNAILLib.ScrollBarsEnum.exDisableBoth;
axThumbnail1.set_ScrollPartVisible(EXTHUMBNAILLib.ScrollBarEnum.exVScroll,
EXTHUMBNAILLib.ScrollPartEnum.exLeftB1Part |
EXTHUMBNAILLib.ScrollPartEnum.exRightB1Part, true);
axThumbnail1.set_ScrollPartCaption(EXTHUMBNAILLib.ScrollBarEnum.exVScroll,
EXTHUMBNAILLib.ScrollPartEnum.exLeftB1Part , "<img></img>1");
axThumbnail1.set_ScrollPartCaption(EXTHUMBNAILLib.ScrollBarEnum.exVScroll,
EXTHUMBNAILLib.ScrollPartEnum.exRightB1Part, "<img></img>2");
axThumbnail1.EndUpdate();
```

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

```
m_thumbnail.BeginUpdate();
m_thumbnail.SetScrollBars( 15 /*exDisableBoth*/ );
m_thumbnail.SetScrollPartVisible( 0 /*exVScroll*/, 32768  /*exLeftB1Part*/ | 32
/*exRightB1Part*/, TRUE );
m_thumbnail.SetScrollPartCaption( 0 /*exVScroll*/, 32768  /*exLeftB1Part*/ , _T("<img>
</img>1") );
m_thumbnail.SetScrollPartCaption( 0 /*exVScroll*/, 32 /*exRightB1Part*/ , _T("<img>
</img>2") );
m_thumbnail.EndUpdate();
```

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

```
With thisform.Thumbnail1
   .BeginUpdate
      .ScrollBars = 15
      .ScrollPartVisible(0, bitor(32768,32)) = .t.
      .ScrollPartCaption(0,32768) = "<img></img>1"
      .ScrollPartCaption(0, 32) = "<img></img>2"
   .EndUpdate
EndWith
```

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

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

property Thumbnail.ScrollThumbSize(ScrollBar as ScrollBarEnum) as Long

Specifies the size of the thumb in the scrollbar.

Type	Description
ScrollBar as ScrollBarEnum	A ScrollBarEnum expression that indicates the vertical or the horizontal scroll bar.
Long	A long expression that defines the size of the scrollbar's thumb.

Use the ScrollThumbSize property to define a fixed size for the scrollbar's thumb. By default, the ScrollThumbSize property is -1, that makes the control computes automatically the size of the thumb based on the scrollbar's range. If case, use the fixed size for your thumb when you change its visual appearance using the Background(exVSThumb) or Background(exHSThumb) property. Use the ScrollWidth property to specify the width of the vertical scroll bar. Use the ScrollButtonWidth property to specify the width of the buttons in the horizontal scroll bar. Use the ScrollHeight property to specify the height of the horizontal scroll bar. Use the ScrollButtonHeight property to specify the height of the buttons in the vertical scroll bar. Use the ScrollPartVisible property to specify the visible parts in the control's scroll bar.

property Thumbnail.ScrollToolTip(ScrollBar as ScrollBarEnum) as String

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

Type	Description
ScrollBar as ScrollBarEnum	A ScrollBarEnum expression that indicates the vertical scroll bar or the horizontal scroll bar.
String	A string expression being shown when the user clicks and moves the scrollbar's thumb.

Use the ScrollToolTip property to specify whether the control displays a tooltip when the user clicks and moves the scrollbar's thumb. By default, the ScrollToolTip property is empty. If the ScrollToolTip property is empty, the tooltip is not shown when the user clicks and moves the thumb of the scroll bar. Use the SortPartVisible property to specify the parts being visible in the control's scroll bar.

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

```
Private Sub Thumbnail1_OffsetChanged(ByVal Horizontal As Boolean, ByVal NewVal As
Long)
   If (Not Horizontal) Then
      Thumbnail1.ScrollToolTip(exVScroll) = "Record " & NewVal
   End If
End Sub
```

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

```
Private Sub AxThumbnail1_OffsetChanged(ByVal sender As System.Object, ByVal e As
AxEXTHUMBNAILLib._IThumbnailEvents_OffsetChangedEvent) Handles
AxThumbnail1.OffsetChanged
   If (Not e.horizontal) Then
      AxThumbnail1.set_ScrollToolTip(EXTHUMBNAILLib.ScrollBarEnum.exVScroll, "Record "
& e.newVal.ToString())
   End If
End Sub
```

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

```
void OnOffsetChangedThumbnail1(BOOL Horizontal, long NewVal)
{
   if ( !Horizontal )
   {
      CString strFormat;
      strFormat.Format( _T("%i"), NewVal );
      m_thumbnail.SetScrollToolTip( 0, strFormat );
   }
}
```

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

```
private void axThumbnail1_OffsetChanged(object sender,
AxEXTHUMBNAILLib._IThumbnailEvents_OffsetChangedEvent e)
{
   if ( !e.horizontal )
      axThumbnail1.set_ScrollToolTip(EXTHUMBNAILLib.ScrollBarEnum.exVScroll, "Record "
+ e.newVal.ToString());
}
```

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

```
*** ActiveX Control Event ***
LPARAMETERS horizontal, newval

If (1 # horizontal) Then
   thisform.Thumbnail1.ScrollToolTip(0) = "Record " + ltrim(str(newval))
EndIf
```

property Thumbnail.ScrollWidth as Long

Specifies the width of the vertical scrollbar.

Type	Description
Long	A long expression that defines the width of the vertical scroll bar.

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

property Thumbnail.Select as String

The Select property indicates the name of the selected thumbnail.

Type	Description
String	A String expression that specifies the full name of the selected file.

The Select property indicates the name of the selected thumbnail. You can use the Select property to programmatically selects a file by its name or full name. The StateChange(SelChangeState) event notifies your application once the user selects a new file within the control. The Select property of the control gets/sets all selected thumbnail(s) separated by "\r\n" (vbCrLf) sequence, while SingleSel property is False. The SingleSel property Retrieves or sets a value that indicates whether the control supports single or multiple selection.

You can use any of the following properties/method to add files into the control:

- InputFile property specifies the file or the web page to display its thumbnail.
- InputFiles property indicates the list of files to be thumbnailed
- AddInputFiles method adds files to be thumbnailed.

property Thumbnail.ShowContextMenu as String

Specifies the object's context menu.

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

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

For instance:

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

The ShowContextMenu property supports the following predefined keywords:

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

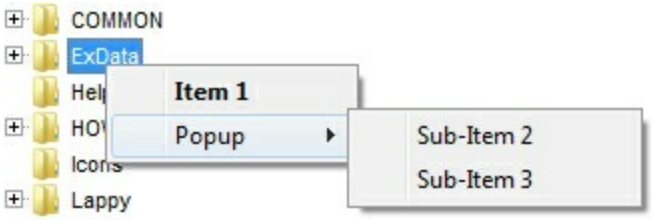
1)

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

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

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

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



property Thumbnail.ShowImageList as Boolean

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

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

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

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

Shows the specified tooltip at given position.

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

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

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

For instance:

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

to its default position

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

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

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

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

gradient-center

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

height of the font. For instance the "<**out** 000000><fgcolor=FFFFFF>outlined</fgcolor></**out**>" generates the following picture:

- **<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 Thumbnail.SingleCaption as String

Indicates the expression to generate the HTML caption to be displayed on the thumbnail, when the control shows a single thumbnail.

Type	Description
String	A String value that defines the expression to generate the HTML caption to be displayed on the thumbnail, when the control shows a single thumbnail.

By default, the SingleCaption property is "ffolder ? (`Name: <b>` + fname + `</b>` + `<br>` + (len(0:= ftype) ? `Type: <b>` + =:0 + `</b>` + `<br>` : `` ) + ( ffolder = 1 ? `Size: <b>` + fsizeF + `</b>` + `<br>` : ``) + `Created: <b>` + date(fcreated - bias/24/60) + `</b>` + `<br>` + `Modified: <b>` + date(fmodified - bias/24/60) + `</b>`+ `<br>` + `Last Opened: <b>` + date(fopened - bias/24/60) + `</b>`+ (fpicture ? (`<br>Dimensions: <b>` + fwidth + ` x ` + fheight + `</b>`) : ``)) : ffile" which makes the control to display full information about the file like seen in the following screen shot. The SingleCaption property indicates the expression to generate the HTML caption to be displayed on the thumbnail, when the control shows a single thumbnail (for instance you filter and you got a single result). Use the Caption property to specify a caption to be displayed on the thumbnail.

Use the Alignment property to specify the caption's alignment. The CaptionRotate property rotates the HTML caption. Use the Transparency property to specify the transparency to display the caption on the thumbnail control. Use the Font property to specify the font to display the caption on the thumbnail. Use the ForeColor property to specify the control's foreground color. Use the WordWrap property to wrap the caption on the control. The

StatusCaption property indicates the expression to generate the HTML caption to be displayed on the thumbnail's status bar.

For instance:

- "ffile" displays the full name of the file
- "`Name: <b>` + fname", displays just the name of the file
- "`Created: <b>` + longdate(fcreated) + ` ` + time(fcreated)", displays the date-time the file was created

The SingleCaption property supports the following keywords:

- **fcount** keyword, returns the number of thumbnail files
- **findex** keyword, specifies the index of the thumbnail file
- **width** keyword, specifies the current width of the thumbnail view
- **height** keyword, specifies the current height of the thumbnail view
- **ffile** keyword, returns the file
- **fname** keyword, returns the name of the file
- **faname** keyword, returns the alternate name of the file, 8.3 format truncated version of the name
- **fext** keyword, returns the extension of the file
- **fcreated** keyword, specifies when a file or directory was created
- **fmodified** keyword, specifies when a file or directory was last modified
- **fopened** keyword, specifies when a file or directory was last accessed
- **fsize** keyword, specifies the size of the file
- **fsizeF** keyword, returns automatically the size of the file in KB, MB or GB
- **ffolder** keyword, returns 0, 1, 2, -1 or -2, if the object does not exist, it is a file, a shell item, a folder or a drive
- **fattr** keyword, returns the attributes of the file/folder
- **ftype** keyword, returns the type of the file/folder, like JPG File, ...
- **fpicture** keyword, indicates whether the file points to a known picture type, like BMP, PNG, and so on
- **fwidth** keyword, returns the width of the picture file
- **fheight** keyword, returns the height of the picture file
- **ffilter** keyword, returns the currently filter pattern (FilterBarPromptPattern property)
- **fvcount** keyword, indicates the number of visible thumbnails. For instance, if a filter is applied the fvcount returns the number of available/visible/filtered thumbnails.
- **fvindex** keyword, indicates the index of the thumbnail in the visible collection.

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

The SingleCaption property supports the following HTML tags:

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

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

- **<img>key[:width]</img>** inserts a custom size picture into the text being previously loaded using the HTMLPicture property. The Key parameter indicates the key of the picture being displayed. The Width parameter indicates a custom size, if you require to stretch the picture, else the original size of the picture is used.
- **&** glyph characters as **&** (&), **<** (<), **>** (>), **&qout;** (") and **&#number;** (the character with specified code), For instance, the € displays the EUR character. The **&** ampersand is only recognized as markup when it is followed by a known letter or a #character and a digit. For instance if you want to display bold in HTML caption you can use bold
- **<off offset> ... </off>** defines the vertical offset to display the text/element. The offset parameter defines the offset to display the element. This tag is inheritable, so the offset is keep while the associated </off> tag is found. You can use the <off offset> HTML tag in combination with the to define a smaller or a larger font to be displayed. For instance: "Text with <**off** 6>subscript" displays the text such as: Text with $_{\text{subscript}}$ The "Text with <**off** -6>superscript" displays the text such as: Text with ^{subscript}
- **<gra rrggbb;mode;blend> ... </gra>** defines a gradient text. The text color or <fgcolor> defines the starting gradient color, while the rr/gg/bb represents the red/green/blue values of the ending color, 808080 if missing as gray. The mode is a value between 0 and 4, 1 if missing, and blend could be 0 or 1, 0 if missing. The HTML tag can be used to define the height of the font. Any of the rrggbb, mode or blend field may not be specified. The <gra> with no fields, shows a vertical gradient color from the current text color to gray (808080). For instance the "<**gra** FFFFFF;1;1>gradient-center</**gra**>" generates the following picture:

gradient-center

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

outlined

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

shadow

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

outline anti-aliasing

property Thumbnail.SingleSel as Boolean

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

Type	Description
Boolean	A Boolean expression that specifies whether the user can select a single thumbnail or more.

By default, the SingleSel property is true, which indicates that only one thumbnail can be selected at any time. The SingleSel property Retrieves or sets a value that indicates whether the control supports single or multiple selection. The Select property gets or sets the file(s)/thumbnail(s) selected. If SingleSel property is False, the user can select mutliple-thumbnail(s) using the CTRL key.

property Thumbnail.Sort as ThumbnailSortEnum

Sorts the thumbnails.

Type	Description
ThumbnailSortEnum	A ThumbnailSortEnum expression that indicates how the files are being sorted.

By default the Sort property is exThumbailUnsorted, which indicates that the files are being displayed as being loaded. You can use the Sort property to programmatically sort the files within the control. Use the AcceptFiles property to disable drag-and-drop files in the thumbnail control. The AcceptFolders property specifies whether the control accepts drag-and-drop folders. The LimitInputFiles property limits the number of files the control can display. You can use the exThumbailSortReverse flag to reverse the sorting order.

You can use any of the following properties/method to add files into the control:

- InputFile property specifies the file or the web page to display its thumbnail.
- InputFiles property indicates the list of files to be thumbnailed
- AddInputFiles method adds files to be thumbnailed.

property Thumbnail.StatusCaption as String

Indicates the expression to generate the HTML caption to be displayed on the thumbnail's status bar.

Type	Description
String	A String expression that defines the expression to generate the HTML caption to be displayed on the thumbnail's status bar.

By default, the StatusCaption property is empty. The StatusCaption property indicates the expression to generate the HTML caption to be displayed on the thumbnail's status bar. Use the Caption property to specify a caption to be displayed on the thumbnail. The SingleCaption property indicates the expression to generate the HTML caption to be displayed on the thumbnail, when the control shows a single thumbnail (for instance you filter and you got a single result).

For instance:

- "`<r><font ;6><fgcolor=808080><sha ;;0>Count: <b>` + (len(fvcount) ? fvcount : 0)" displays the number of files being shown in the list.

property Thumbnail.Template as String

Specifies the control's template.

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

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

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

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

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

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

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

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

of the object in the context. The "list or arguments" may include variables or values separated by commas. (Sample: h = InsertItem(0,"New Child"))

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

The x-script may uses constant expressions as follow:

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

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

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

property Thumbnail.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 Thumbnail.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 Thumbnail.Thumbnail as IPictureDisp

Retrieves the thumbnail view of the file.

Type	Description
IPictureDisp	A Picture object that holds the BMP or ICO representation of the thumbnail.

Use the Thumbnail property to access the thumbnail picture. Use the InputFile property to specify the file whose thumbnail is retrieved. Use the ThumbnailType property to specify whether the thumbnail is a bitmap or an icon. For instance, if you set the ThumbnailType property on exThumbnailIcon, and calling the InputFile property, the control gets the icon representation of the file. If the ThumbnailType property on exThumbnailIcon is exThumbnailBitmap the control tries to get a bitmap representation of the file if it is available, else the Thumbnail property is nothing, and the ThumbnailType property is exNoThumbnail. Use the OutputFile property to save the thumbnail picture. The Bitmap property always returns a BITMAP object that shows the file's thumbnail.

Let's say that we drop the floppy.png file to the component, and based on the ThumbnailType property the component generates the following views:

1. ***exThumbnailPicture*** *(*The thumbnail type is a bitmap, but the original file is a picture being loaded directly*)*

2. ***exThumbnailBitmap*** *(*The thumbnail type is a bitmap*)*

3. ***exThumbnailIcon*** *(*The thumbnail type is an icon.*)*

property Thumbnail.ThumbnailFromPoint (X as OLE_XPOS_PIXELS, Y as OLE_YPOS_PIXELS) as Variant

Retrieves the thumbnail from point.

Type	Description
X as OLE_XPOS_PIXELS	A single that specifies the current X location of the mouse pointer. The x values is always expressed in client coordinates
Y as OLE_YPOS_PIXELS	A single that specifies the current X location of the mouse pointer. The x values is always expressed in client coordinates.
Variant	A String expression that specifies the full name of the file from the specified location.

The ThumbnailFromPoint(-1,-1) property returns the full name of the file from the current cursor location. The Select property selects programmatically a file by its name or full name.

property Thumbnail.ThumbnailHeight as Long

Specifies the height to display the thumbnails.

Type	Description
Long	A long expression that specifies height in pixels of each thumbnail.

At runtime, the user can resize the thumbnail, by click and drag the middle mouse button or by double-clicking the thumbnail. Excludes the exThumbnailAllowResize flag from the Mode property to disable resizing the thumbnails when the user click and drag the middle mouse button. Excludes the exThumbnailAutoFitOnDblClk flag from the Mode property to prevent enlarging the thumbnail when user double-clicks it. The exThumbnailKeepAspectRatio flag of the Mode property specifies whether the thumbnail is keeping its aspect ratio when it is displayed in the control. The exThumbnailStretch flag of the Mode property stretches the thumbnail on the frame's client area.

You can use the following properties to define the distance and padding of the thumbnails

- The Borders property indicates the margins where the thumbnail is displayed, relative to the control's border.
- The Margins property specifies the distance between thumbnails, when multiple thumbnails are displayed on the control.
- The Padding property generates space around thumbnail. Clears the area around the content (inside the border) of the thumbnail.

The following properties specifies the size of each thumbnail:

- The ThumbnailWidth property specifies the width to display the thumbnails.
- The ThumbnailHeight property specifies the height to display the thumbnails.

The following properties specifies the limit / range of the size for of each thumbnail:

- The ThumbnailMinWidth property specifies the minimum width to display the thumbnails.
- The ThumbnailMaxWidth property specifies the maximum width to display the thumbnails.
- The ThumbnailMinHeight property specifies the minimum height to display the thumbnails.
- The ThumbnailMaxHeight property specifies the maximum height to display the thumbnails.

property Thumbnail.ThumbnailMaxHeight as Long

Specifies the maximum height to display the thumbnails.

Type	Description
Long	A long expression that specifies the maximum height to display the thumbnails.

At runtime, the user can resize the thumbnail, by click and drag the middle mouse button or by double-clicking the thumbnail. Excludes the exThumbnailAllowResize flag from the Mode property to disable resizing the thumbnails when the user click and drag the middle mouse button. Excludes the exThumbnailAutoFitOnDblClk flag from the Mode property to prevent enlarging the thumbnail when user double-clicks it. The exThumbnailKeepAspectRatio flag of the Mode property specifies whether the thumbnail is keeping its aspect ratio when it is displayed in the control. The exThumbnailStretch flag of the Mode property stretches the thumbnail on the frame's client area.

You can use the following properties to define the distance and padding of the thumbnails

- The Borders property indicates the margins where the thumbnail is displayed, relative to the control's border.
- The Margins property specifies the distance between thumbnails, when multiple thumbnails are displayed on the control.
- The Padding property generates space around thumbnail. Clears the area around the content (inside the border) of the thumbnail.

The following properties specifies the size of each thumbnail:

- The ThumbnailWidth property specifies the width to display the thumbnails.
- The ThumbnailHeight property specifies the height to display the thumbnails.

The following properties specifies the limit / range of the size for of each thumbnail:

- The ThumbnailMinWidth property specifies the minimum width to display the thumbnails.
- The ThumbnailMaxWidth property specifies the maximum width to display the thumbnails.
- The ThumbnailMinHeight property specifies the minimum height to display the thumbnails.
- The ThumbnailMaxHeight property specifies the maximum height to display the thumbnails.

property Thumbnail.ThumbnailMaxWidth as Long

Specifies the maximum width to display the thumbnails.

Type	Description
Long	A long expression that specifies the maximum width to display the thumbnails.

At runtime, the user can resize the thumbnail, by click and drag the middle mouse button or by double-clicking the thumbnail. Excludes the exThumbnailAllowResize flag from the Mode property to disable resizing the thumbnails when the user click and drag the middle mouse button. Excludes the exThumbnailAutoFitOnDblClk flag from the Mode property to prevent enlarging the thumbnail when user double-clicks it. The exThumbnailKeepAspectRatio flag of the Mode property specifies whether the thumbnail is keeping its aspect ratio when it is displayed in the control. The exThumbnailStretch flag of the Mode property stretches the thumbnail on the frame's client area.

You can use the following properties to define the distance and padding of the thumbnails

- The Borders property indicates the margins where the thumbnail is displayed, relative to the control's border.
- The Margins property specifies the distance between thumbnails, when multiple thumbnails are displayed on the control.
- The Padding property generates space around thumbnail. Clears the area around the content (inside the border) of the thumbnail.

The following properties specifies the size of each thumbnail:

- The ThumbnailWidth property specifies the width to display the thumbnails.
- The ThumbnailHeight property specifies the height to display the thumbnails.

The following properties specifies the limit / range of the size for of each thumbnail:

- The ThumbnailMinWidth property specifies the minimum width to display the thumbnails.
- The ThumbnailMaxWidth property specifies the maximum width to display the thumbnails.
- The ThumbnailMinHeight property specifies the minimum height to display the thumbnails.
- The ThumbnailMaxHeight property specifies the maximum height to display the thumbnails.

property Thumbnail.ThumbnailMinHeight as Long

Specifies the minimum height to display the thumbnails.

Type	Description
Long	A long expression that specifies the minimum height to display the thumbnails.

At runtime, the user can resize the thumbnail, by click and drag the middle mouse button or by double-clicking the thumbnail. Excludes the exThumbnailAllowResize flag from the Mode property to disable resizing the thumbnails when the user click and drag the middle mouse button. Excludes the exThumbnailAutoFitOnDblClk flag from the Mode property to prevent enlarging the thumbnail when user double-clicks it. The exThumbnailKeepAspectRatio flag of the Mode property specifies whether the thumbnail is keeping its aspect ratio when it is displayed in the control. The exThumbnailStretch flag of the Mode property stretches the thumbnail on the frame's client area.

You can use the following properties to define the distance and padding of the thumbnails

- The Borders property indicates the margins where the thumbnail is displayed, relative to the control's border.
- The Margins property specifies the distance between thumbnails, when multiple thumbnails are displayed on the control.
- The Padding property generates space around thumbnail. Clears the area around the content (inside the border) of the thumbnail.

The following properties specifies the size of each thumbnail:

- The ThumbnailWidth property specifies the width to display the thumbnails.
- The ThumbnailHeight property specifies the height to display the thumbnails.

The following properties specifies the limit / range of the size for of each thumbnail:

- The ThumbnailMinWidth property specifies the minimum width to display the thumbnails.
- The ThumbnailMaxWidth property specifies the maximum width to display the thumbnails.
- The ThumbnailMinHeight property specifies the minimum height to display the thumbnails.
- The ThumbnailMaxHeight property specifies the maximum height to display the thumbnails.

property Thumbnail.ThumbnailMinWidth as Long

Specifies the minimum width to display the thumbnails.

Type	Description
Long	A long expression that specifies the minimum width to display the thumbnails.

At runtime, the user can resize the thumbnail, by click and drag the middle mouse button or by double-clicking the thumbnail. Excludes the exThumbnailAllowResize flag from the Mode property to disable resizing the thumbnails when the user click and drag the middle mouse button. Excludes the exThumbnailAutoFitOnDblClk flag from the Mode property to prevent enlarging the thumbnail when user double-clicks it. The exThumbnailKeepAspectRatio flag of the Mode property specifies whether the thumbnail is keeping its aspect ratio when it is displayed in the control. The exThumbnailStretch flag of the Mode property stretches the thumbnail on the frame's client area.

You can use the following properties to define the distance and padding of the thumbnails

- The Borders property indicates the margins where the thumbnail is displayed, relative to the control's border.
- The Margins property specifies the distance between thumbnails, when multiple thumbnails are displayed on the control.
- The Padding property generates space around thumbnail. Clears the area around the content (inside the border) of the thumbnail.

The following properties specifies the size of each thumbnail:

- The ThumbnailWidth property specifies the width to display the thumbnails.
- The ThumbnailHeight property specifies the height to display the thumbnails.

The following properties specifies the limit / range of the size for of each thumbnail:

- The ThumbnailMinWidth property specifies the minimum width to display the thumbnails.
- The ThumbnailMaxWidth property specifies the maximum width to display the thumbnails.
- The ThumbnailMinHeight property specifies the minimum height to display the thumbnails.
- The ThumbnailMaxHeight property specifies the maximum height to display the thumbnails.

property Thumbnail.ThumbnailType as ThumbnailTypeEnum

Retrieves the type of the thumbnail view.

Type	Description
ThumbnailTypeEnum	A ThumbnailTypeEnum expression that indicates the type of the thumbnail being loaded.

By default, the ThumbnailType property is exThumbnailAvail.

If the ThumbnailType property is exThumbnailAvail the control checks:

- if the InputFile property points to a picture file, loads directly the picture from, and so the ThumbnailType property retrieves the exThumbnailPicture
- else, if the InputFile has a bitmap representation, the the ThumbnailType property retrieves the exThumbnailBitmap
- else, the ThumbnailType property retrieves the exThumbnailIcon

Use the ThumbnailType property to specify whether the thumbnail is a picture, bitmap or an icon. Use the Thumbnail property to retrieve the thumbnail picture. Use the InputFile property to programmatically display a thumbnail for a specified file. Use the OutputFile property to save the thumbnail picture. For instance, if you set the ThumbnailType property on exThumbnailIcon, and calling the InputFile property, the control gets the icon representation of the file. If the ThumbnailType property on exThumbnailIcon is exThumbnailBitmap the control tries to get a bitmap representation of the file if it is available, else the Thumbnail property is nothing, and the ThumbnailType property is exNoThumbnail.

Let's say that we drop the floppy.png file to the component, and based on the ThumbnailType property the component generates the following views:

1. ***exThumbnailPicture*** *(*The thumbnail type is a bitmap, but the original file is a picture being loaded directly*)*

2. ***exThumbnailBitmap*** *(*The thumbnail type is a bitmap*)*

3. ***exThumbnailIcon*** *(*The thumbnail type is an icon.*)*

property Thumbnail.ThumbnailWidth as Long

Specifies the width to display the thumbnails.

Type	Description
Long	A long expression that specifies width in pixels of each thumbnail.

At runtime, the user can resize the thumbnail, by click and drag the middle mouse button or by double-clicking the thumbnail. Excludes the exThumbnailAllowResize flag from the Mode property to disable resizing the thumbnails when the user click and drag the middle mouse button. Excludes the exThumbnailAutoFitOnDblClk flag from the Mode property to prevent enlarging the thumbnail when user double-clicks it. The exThumbnailKeepAspectRatio flag of the Mode property specifies whether the thumbnail is keeping its aspect ratio when it is displayed in the control. The exThumbnailStretch flag of the Mode property stretches the thumbnail on the frame's client area.

You can use the following properties to define the distance and padding of the thumbnails

- The Borders property indicates the margins where the thumbnail is displayed, relative to the control's border.
- The Margins property specifies the distance between thumbnails, when multiple thumbnails are displayed on the control.
- The Padding property generates space around thumbnail. Clears the area around the content (inside the border) of the thumbnail.

The following properties specifies the size of each thumbnail:

- The ThumbnailWidth property specifies the width to display the thumbnails.
- The ThumbnailHeight property specifies the height to display the thumbnails.

The following properties specifies the limit / range of the size for of each thumbnail:

- The ThumbnailMinWidth property specifies the minimum width to display the thumbnails.
- The ThumbnailMaxWidth property specifies the maximum width to display the thumbnails.
- The ThumbnailMinHeight property specifies the minimum height to display the thumbnails.
- The ThumbnailMaxHeight property specifies the maximum height to display the thumbnails.

property Thumbnail.Timeout as Long

Specifies a value that indicates the number of milliseconds to extract the object's thumbnail.

Type	Description
Long	A long expression that specifies the the number of milliseconds to extract the object's thumbnail.

By default, the Timeout property is set to 2 seconds. If extracting the object's thumbnail takes more that 2 seconds, the icon representation is displayed instead, if available.

property Thumbnail.ToolTipDelay as Long

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

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

If the ToolTipDelay or ToolTipPopDelay property is 0, the control displays no tooltips. The ToolTipPopDelay property specifies the period in ms of time the ToolTip remains visible if the mouse pointer is stationary within a control. Use the ToolTipWidth property to specify the width of the tooltip window. Use the ToolTipText property to specify the control's tooltip. Use the ShowToolTip method to display a custom tooltip.

property Thumbnail.ToolTipFont as IFontDisp

Retrieves or sets the tooltip's font.

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

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

property Thumbnail.ToolTipPopDelay as Long

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

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

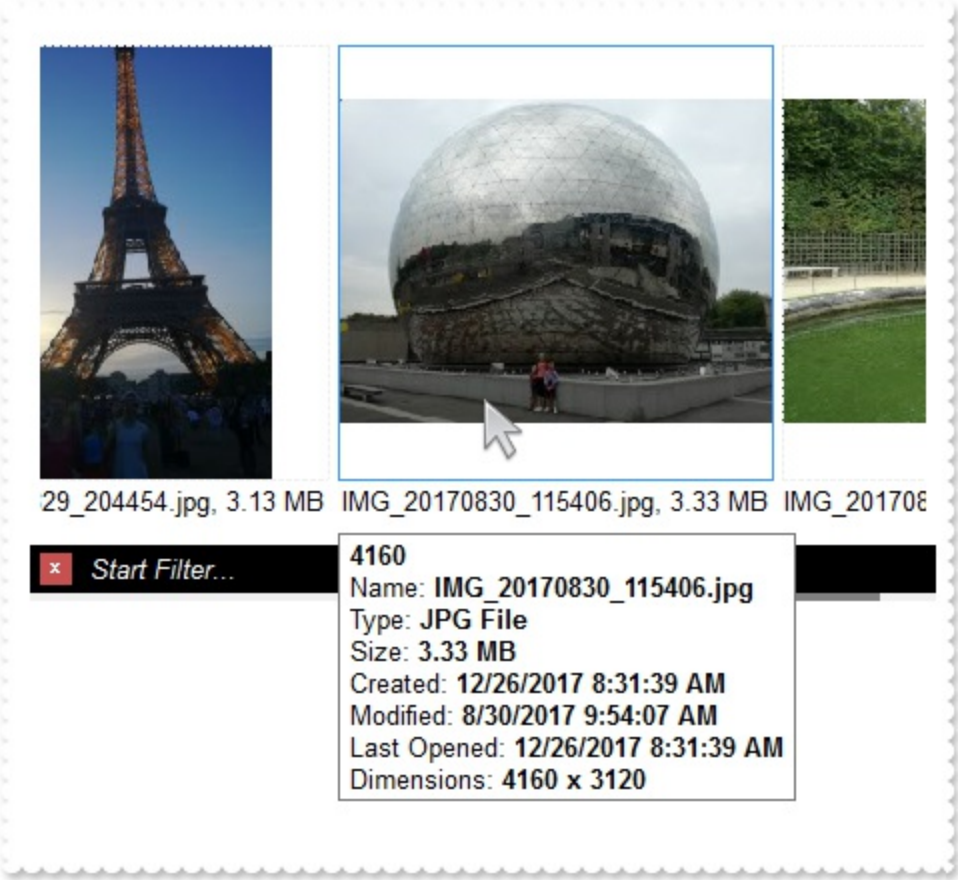
If the ToolTipDelay or ToolTipPopDelay property is 0, the control displays no tooltips. The ToolTipDelay property specifies the time in ms that passes before the ToolTip appears. Use the ToolTipWidth property to specify the width of the tooltip window. Use the ToolTipFont property to assign a font for the control's tooltip. Use the ToolTipText property to specify the control's tooltip. Use the ShowToolTip method to display a custom tooltip.

property Thumbnail.ToolTipText as String

Gets or sets the ToolTip text associated with the thumbnail control.

Type	Description
String	A String expression that specifies the tooltip being shown when user hovers the control.

By default, the ToolTipText property is "ffolder ? (`Name: <b>` + fname + `</b>` + `<br>` + (len(0:= ftype) ? `Type: <b>` + =:0 + `</b>` + `<br>` : `` ) + ( ffolder = 1 ? `Size: <b>` + fsizeF + `</b>` + `<br>` : ``) + `Created: <b>` + date(fcreated - bias/24/60) + `</b>` + `<br>` + `Modified: <b>` + date(fmodified - bias/24/60) + `</b>`+ `<br>` + `Last Opened: <b>` + date(fopened - bias/24/60) + `</b>`+ (fpicture ? (`<br>Dimensions: <b>` + fwidth + ` x ` + fheight + `</b>`) : ``)) : ffile". Use the ToolTipText property to display a tooltip when the cursor hovers the control. Use the ToolTipTitle property to specify the title of the tooltip. If the ToolTipDelay or ToolTipPopDelay property is 0, the control displays no tooltips. The ToolTipPopDelay property specifies the period in ms of time the ToolTip remains visible if the mouse pointer is stationary within a control. Use the ToolTipWidth property to specify the width of the tooltip window. Use the ShowToolTip method to display a custom tooltip. Use the Caption property to specify a caption to be displayed on the thumbnail. The SingleCaption property indicates the expression to generate the HTML caption to be displayed on the thumbnail, when the control shows a single thumbnail (for instance you filter and you got a single result).



The ToolTipText property supports the following keywords:

- **fcount** keyword, returns the number of thumbnail files
- **findex** keyword, specifies the index of the thumbnail file
- **width** keyword, specifies the current width of the thumbnail view
- **height** keyword, specifies the current height of the thumbnail view
- **ffile** keyword, returns the file
- **fname** keyword, returns the name of the file
- **faname** keyword, returns the alternate name of the file, 8.3 format truncated version of the name
- **fext** keyword, returns the extension of the file
- **fcreated** keyword, specifies when a file or directory was created (UTC)
- **fcreated_local** keyword, specifies when a file or directory was created (local date-time)
- **fmodified** keyword, specifies when a file or directory was last modified (UTC)
- **fmodified_local** keyword, specifies when a file or directory was last modified (local date-time)
- **fopened** keyword, specifies when a file or directory was last accessed (UTC)
- **fopened_local** keyword, specifies when a file or directory was last accessed (local date-time)
- **fsize** keyword, specifies the size of the file
- **fsizeF** keyword, returns automatically the size of the file in KB, MB or GB
- **ffolder** keyword, returns 0, 1, 2, -1 or -2, if the object does not exist, it is a file, a shell item, a folder or a drive
- **fattr** keyword, returns the attributes of the file/folder
- **ftype** keyword, returns the type of the file/folder, like JPG File, ...
- **fpicture** keyword, indicates whether the file points to a known picture type, like BMP, PNG, and so on
- **fwidth** keyword, returns the width of the picture file
- **fheight** keyword, returns the height of the picture file
- **ffilter** keyword, returns the currently filter pattern (FilterBarPromptPattern property)
- **fvcount** keyword, indicates the number of visible thumbnails. For instance, if a filter is applied the fvcount returns the number of available/visible/filtered thumbnails.
- **fvindex** keyword, indicates the index of the thumbnail in the visible collection.

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

The ToolTipText property supports the following HTML tags:

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

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

picture being displayed. The Width parameter indicates a custom size, if you require to stretch the picture, else the original size of the picture is used.

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

gradient-center

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

outlined

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

shadow

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

property Thumbnail.ToolTipTitle as String

Gets or sets the ToolTip title associated with the thumbnail control.

Type	Description
String	A string expression that specifies the title of the tooltip.

By default, the ToolTipTitle property is empty. Use the ToolTipText property to display a tooltip when the cursor hovers the control. If the ToolTipDelay or ToolTipPopDelay property is 0, the control displays no tooltips. The ToolTipPopDelay property specifies the period in ms of time the ToolTip remains visible if the mouse pointer is stationary within a control. Use the ToolTipWidth property to specify the width of the tooltip window. Use the ShowToolTip method to display a custom tooltip.

The ToolTipTitle property supports the following keywords:

- **fcount** keyword, returns the number of thumbnail files
- **findex** keyword, specifies the index of the thumbnail file
- **width** keyword, specifies the current width of the thumbnail view
- **height** keyword, specifies the current height of the thumbnail view
- **ffile** keyword, returns the file
- **fname** keyword, returns the name of the file
- **faname** keyword, returns the alternate name of the file, 8.3 format truncated version of the name
- **fext** keyword, returns the extension of the file
- **fcreated** keyword, specifies when a file or directory was created (UTC)
- **fcreated_local** keyword, specifies when a file or directory was created (local date-time)
- **fmodified** keyword, specifies when a file or directory was last modified (UTC)
- **fmodified_local** keyword, specifies when a file or directory was last modified (local date-time)
- **fopened** keyword, specifies when a file or directory was last accessed (UTC)
- **fopened_local** keyword, specifies when a file or directory was last accessed (local date-time)
- **fsize** keyword, specifies the size of the file
- **fsizeF** keyword, returns automatically the size of the file in KB, MB or GB
- **ffolder** keyword, returns 0, 1, 2, -1 or -2, if the object does not exist, it is a file, a shell item, a folder or a drive
- **fattr** keyword, returns the attributes of the file/folder
- **ftype** keyword, returns the type of the file/folder, like JPG File, ...
- **fpicture** keyword, indicates whether the file points to a known picture type, like BMP, PNG, and so on
- **fwidth** keyword, returns the width of the picture file

- **fheight** keyword, returns the height of the picture file
- **ffilter** keyword, returns the currently filter pattern (FilterBarPromptPattern property)
- **fvcount** keyword, indicates the number of visible thumbnails. For instance, if a filter is applied the fvcount returns the number of available/visible/filtered thumbnails.
- **fvindex** keyword, indicates the index of the thumbnail in the visible collection.

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

property Thumbnail.ToolTipWidth as Long

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

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

Use the ToolTipWidth property to change the tooltip window width. The height of the tooltip window is automatically computed based on tooltip's description. The ToolTipPopDelay property specifies the period in ms of time the ToolTip remains visible if the mouse pointer is stationary within a control. The ToolTipDelay property specifies the time in ms that passes before the ToolTip appears. Use the ToolTipFont property to assign a font for the control's tooltip. Use the ToolTipText property to specify the control's tooltip. Use the ShowToolTip method to display a custom tooltip.

property Thumbnail.Transparency as Long

Specifies the transparency to display the text in the control.

Type	Description
Long	A long expression that specifies the percent of transparency used to paint the caption. The value should be from 0 to 100, where 0 means opaque.

By default, the Transparency property is 0, which means that the caption is opaque. Use the Caption property to display a caption on the thumbnail control. The SingleCaption property indicates the expression to generate the HTML caption to be displayed on the thumbnail, when the control shows a single thumbnail (for instance you filter and you got a single result). Use the InputFile property to specify the file to display its thumbnail. Use the OutputFile property to save the thumbnail picture. Use the Font property to specify the font to display the caption on the thumbnail. Use the ForeColor property to specify the control's foreground color. Use the WordWrap property to wrap the caption on the control.

property Thumbnail.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 Thumbnail.VirtualMode as Boolean

Specifies whether the control is running in virtual mode.

Type	Description
Boolean	A boolean expression that indicates whether the control is running in virtual mode.

By default, the VirtualMode property is True. The VirtualMode property specifies whether the control is running in virtual mode. When the VirtualModex property is True, the thumbnails are generated for the files being visible only. When the VirtualMode property is False, the control loads in the background the thumbnails for all files. The AutoUpdate property indicates whether the control automatically updates the thumbnail's view being changed externally. Once the AutoUpdate property is True, the control automatically regenerates the thumbnails for files being changed in the current view.

property Thumbnail.WordWrap as Boolean

Indicates whether a multiline label control automatically wraps words to the beginning of the next line when necessary.

Type	Description
Boolean	A boolean expression that specifies whether the control displays its caption using multiple lines.

By default, the WordWrap property is False. Use the WordWrap property to specify whether the control's caption is displayed on multiple lines. Use the Caption property to display a caption in the thumbnail control. The SingleCaption property indicates the expression to generate the HTML caption to be displayed on the thumbnail, when the control shows a single thumbnail (for instance you filter and you got a single result). Use the Alignment property to specify the caption's alignment. Use the Boders property to specify the size of the margins where the thumbnail is displayed. Use the Transparency property to specify the transparency to display the caption on the thumbnail control.

ExThumbnail 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: {44BA596F-E225-476B-81B1-3BDE56AC595A}. The object's program identifier is: "Exontrol.Thumbnail". The /COM object module is: "ExThumbnail.dll"

The ExThumbnail component supports the following events:

Name	Description
AnchorClick	Occurs when an anchor element is clicked.
Change	Occurs when the thumbnail view is changed.
Changing	Occurs when the thumbnail view is changing.
Click	Occurs when the user presses and then releases the left mouse button over the control.
DblClick	Occurs when the user dblclk the left mouse button over an object.
Event	Notifies the application once the control fires an event.
KeyDown	Occurs when the user presses a key while an object has the focus.
KeyPress	Occurs when the user presses and releases an ANSI key.
KeyUp	Occurs when the user releases a key while an object has the focus.
MouseDown	Occurs when the user presses a mouse button.
MouseMove	Occurs when the user moves the mouse.
MouseUp	Occurs when the user releases a mouse button.
OLECompleteDrag	Occurs when a source component is dropped onto a target component, informing the source component that a drag action was either performed or canceled
OLEDragDrop	Occurs when a source component is dropped onto a target component when the source component determines that a drop can occur.
OLEDragOver	Occurs when one component is dragged over another.
OLEGiveFeedback	Allows the drag source to specify the type of OLE drag-and-drop operation and the visual feedback.
OLESetData	Occurs on a drag source when a drop target calls the GetData method and there is no data in a specified format in the OLE drag-and-drop DataObject.

OLEStartDrag	Occurs when the OLEDrag method is called.
StateChange	Fired while the control's state has been changed.

event AnchorClick (AnchorID as String, Options as String)

Occurs when an anchor element is clicked.

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

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

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

C#

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

VB

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

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

C#

```
private void AnchorClick(object sender,
AxEXTHUMBNAILLib._IThumbnailEvents_AnchorClickEvent e)
{
}
```

C++

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

C++ Builder

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

Delphi

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

Delphi 8 (.NET only)

```
procedure AnchorClick(sender: System.Object; e:
AxEXTHUMBNAILLib._IThumbnailEvents_AnchorClickEvent);
begin
end;
```

Powe…

```
begin event AnchorClick(string AnchorID,string Options)
end event AnchorClick
```

VB.NET

```
Private Sub AnchorClick(ByVal sender As System.Object, ByVal e As
AxEXTHUMBNAILLib._IThumbnailEvents_AnchorClickEvent) Handles AnchorClick
End Sub
```

VB6

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

VBA

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

VFP

```
LPARAMETERS AnchorID,Options
```

Xbas…

```
PROCEDURE OnAnchorClick(oThumbnail,AnchorID,Options)
RETURN
```

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

Java…

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

VBSc…

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

Visual Data…

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

Visual Objects

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

X++

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

XBasic

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

dBASE

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

event Change ()

Occurs when the thumbnail view is changed.

Type	Description

The Change event is fired when the user drags a file into the thumbnail control or when the InputFile property is changed. The Changing event is fired before updating the InputFile property. Use the AcceptFiles property to allow user drags files to the thumbnail control. Use the InputFile property to specify the file name or the web page being thumb nailed. Use the Thumbnail property to retrieve the thumbnail picture.

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

C#

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

VB

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

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

C#

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

C++

```
void OnChange()
{
}
```

C++ Builder

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

Delphi

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

Delphi 8 (.NET only)

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

Powe…

```
begin event Change()
end event Change
```

VB.NET

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

VB6

```
Private Sub Change()
End Sub
```

VBA

```
Private Sub Change()
End Sub
```

VFP

```
LPARAMETERS nop
```

Xbas…

```
PROCEDURE OnChange(oThumbnail)
RETURN
```

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

Java…

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

VBSc…

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

Visual Data…

```
Procedure OnComChange
   Forward Send OnComChange
```

```
End_Procedure
```

Visual Objects

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

X++

```
void onEvent_Change()
{
}
```

XBasic

```
function Change as v ()
end function
```

dBASE

```
function nativeObject_Change()
return
```

event Changing (InputFile as String, Cancel as Variant)

Occurs when the thumbnail view is changing.

Type	Description
InputFile as String	A String expression that specifies the file to be previewed.
Cancel as Variant	A Boolean expression that specifies whether the operation is performed or canceled. By default, the Cancel parameter is False, which means the operation is going to be performed.

The control fires the Changing event just before previewing the giving file. In other words, the Changing event is fired when the user drags a file over the control, so you can specify whether to accept or not the file. The control fires the Change event when the previewing is done. Use the Thumbnail property to specify whether the control has a thumbnail view loaded. Change the Cancel parameter to True, if you need to cancel dropping the file. The InputFile property specifies the file being previewed.

The following VB sample prompts whether to accept or reject previewing the dropped file:

```
Private Sub Thumbnail1_Changing(ByVal Path As String, Cancel As Variant)
   Cancel = MsgBox("Do you want to preview the " & Path, vbYesNo) = vbNo
End Sub
```

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

C#

```
private void Changing(object sender,string InputFile,ref object Cancel)
{
}
```

VB

```
Private Sub Changing(ByVal sender As System.Object,ByVal InputFile As
String,ByRef Cancel As Object) Handles Changing
End Sub
```

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

C#

```
private void Changing(object sender,
AxEXTHUMBNAILLib._IThumbnailEvents_ChangingEvent e)
{
}
```

C++

```
void OnChanging(LPCTSTR InputFile,VARIANT FAR* Cancel)
{
}
```

C++ Builder

```
void __fastcall Changing(TObject *Sender,BSTR InputFile,Variant * Cancel)
{
}
```

Delphi

```
procedure Changing(ASender: TObject; InputFile : WideString;var Cancel :
OleVariant);
begin
end;
```

Delphi 8 (.NET only)

```
procedure Changing(sender: System.Object; e:
AxEXTHUMBNAILLib._IThumbnailEvents_ChangingEvent);
begin
end;
```

Powe…

```
begin event Changing(string InputFile,any Cancel)
end event Changing
```

VB.NET

```
Private Sub Changing(ByVal sender As System.Object, ByVal e As
AxEXTHUMBNAILLib._IThumbnailEvents_ChangingEvent) Handles Changing
End Sub
```

VB6

```
Private Sub Changing(ByVal InputFile As String,Cancel As Variant)
End Sub
```

VBA

```
Private Sub Changing(ByVal InputFile As String,Cancel As Variant)
End Sub
```

VFP

```
LPARAMETERS InputFile,Cancel
```

Xbas…

```
PROCEDURE OnChanging(oThumbnail,InputFile,Cancel)
RETURN
```

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

Java…

```
<SCRIPT EVENT="Changing(InputFile,Cancel)" LANGUAGE="JScript">
</SCRIPT>
```

VBSc…

```
<SCRIPT LANGUAGE="VBScript">
Function Changing(InputFile,Cancel)
End Function
</SCRIPT>
```

Visual Data…

```
Procedure OnComChanging String llInputFile Variant llCancel
   Forward Send OnComChanging llInputFile llCancel
End_Procedure
```

Visual Objects

```
METHOD OCX_Changing(InputFile,Cancel) CLASS MainDialog
RETURN NIL
```

X++

```
void onEvent_Changing(str _InputFile,COMVariant /*variant*/ _Cancel)
{
}
```

XBasic

```
function Changing as v (InputFile as C,Cancel as A)
end function
```

dBASE

```
function nativeObject_Changing(InputFile,Cancel)
return
```

event Click ()

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

Type	Description

The Click event is fired when the user releases the left mouse button over the control. 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 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(oThumbnail)
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

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,
AxEXTHUMBNAILLib._IThumbnailEvents_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:
AxEXTHUMBNAILLib._IThumbnailEvents_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
AxEXTHUMBNAILLib._IThumbnailEvents_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(oThumbnail,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.Thumbnail.1::OLE_XPOS_PIXELS,Y as
OLE::Exontrol.Thumbnail.1::OLE_YPOS_PIXELS)
end function
```

dBASE

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

event Event (EventID as Long)

Notifies the application once the control fires an event.

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

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

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

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

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

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

```
// Notifies the application once the control fires an event.
void onEvent_Event(int _EventID)
{
   print thumbnail1.EventParam(-2).toString();
}
```

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

```
OLEGiveFeedback/1004( 1 , =true )
OLEDragDrop/1001( [Object] , =1 , 0 , 0 , 177 , 141 )
OLECompleteDrag/1003( 1 )
```

```
MouseMove/-606( 0 , 0 , 177 , 141 )
MouseMove/-606( 0 , 0 , 178 , 139 )
```

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 be changed during the event handler.

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

C#

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

VB

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

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

C#

```
private void Event(object sender,
AxEXTHUMBNAILLib._IThumbnailEvents_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:
AxEXTHUMBNAILLib._IThumbnailEvents_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
AxEXTHUMBNAILLib._IThumbnailEvents_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(oThumbnail,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,
AxEXTHUMBNAILLib._IThumbnailEvents_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:
AxEXTHUMBNAILLib._IThumbnailEvents_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
AxEXTHUMBNAILLib._IThumbnailEvents_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(oThumbnail,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,
AxEXTHUMBNAILLib._IThumbnailEvents_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:
AxEXTHUMBNAILLib._IThumbnailEvents_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
AxEXTHUMBNAILLib._IThumbnailEvents_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(oThumbnail,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,
AxEXTHUMBNAILLib._IThumbnailEvents_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:
AxEXTHUMBNAILLib._IThumbnailEvents_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
AxEXTHUMBNAILLib._IThumbnailEvents_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(oThumbnail,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.

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,
AxEXTHUMBNAILLib._IThumbnailEvents_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:
AxEXTHUMBNAILLib._IThumbnailEvents_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
AxEXTHUMBNAILLib._IThumbnailEvents_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(oThumbnail,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.Thumbnail.1::OLE_XPOS_PIXELS,Y as
OLE::Exontrol.Thumbnail.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. Use the AnchorFromPoint property to retrieve the anchor from the point.

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

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

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

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

C#
```
private void MouseMoveEvent(object sender,
AxEXTHUMBNAILLib._IThumbnailEvents_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:
AxEXTHUMBNAILLib._IThumbnailEvents_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
AxEXTHUMBNAILLib._IThumbnailEvents_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(oThumbnail,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.Thumbnail.1::OLE_XPOS_PIXELS,Y as
OLE::Exontrol.Thumbnail.1::OLE_YPOS_PIXELS)
end function
```

dBASE

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

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

Occurs when the user releases a mouse button.

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

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

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,
AxEXTHUMBNAILLib._IThumbnailEvents_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:
AxEXTHUMBNAILLib._IThumbnailEvents_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
AxEXTHUMBNAILLib._IThumbnailEvents_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(oThumbnail,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.Thumbnail.1::OLE_XPOS_PIXELS,Y as
OLE::Exontrol.Thumbnail.1::OLE_YPOS_PIXELS)
end function
```

dBASE

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

event OLECompleteDrag (Effect as Long)

Occurs when a source component is dropped onto a target component, informing the source component that a drag action was either performed or canceled

Type	Description
Effect as Long	A long set by the source object identifying the action that has been performed, thus allowing the source to take appropriate action if the component was moved (such as the source deleting data if it is moved from one component to another

The OLECompleteDrag event is the final event to be called in an OLE drag/drop operation. This event informs the source component of the action that was performed when the object was dropped onto the target component. The target sets this value through the effect parameter of the OLEDragDrop event. Based on this, the source can then determine the appropriate action it needs to take. For example, if the object was moved into the target (exDropEffectMove), the source needs to delete the object from itself after the move. The control supports only manual OLE drag and drop events. In order to enable OLE drag and drop feature into control you have to set the OLEDropMode and OLEDrag properties.

The settings for Effect are:

- exOLEDropEffectNone (0), Drop target cannot accept the data, or the drop operation was cancelled
- exOLEDropEffectCopy (1), Drop results in a copy of data from the source to the target. The original data is unaltered by the drag operation.
- exOLEDropEffectMove (2), Drop results in data being moved from drag source to drop source. The drag source should remove the data from itself after the move.

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

C#
```
// OLECompleteDrag event is not supported. Use the
DragEnter,DragLeave,DragOver, DragDrop ... events.
```

VB
```
// OLECompleteDrag event is not supported. Use the
DragEnter,DragLeave,DragOver, DragDrop ... events.
```

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

C#
```
private void OLECompleteDrag(object sender,
AxEXTHUMBNAILLib._IThumbnailEvents_OLECompleteDragEvent e)
{
```

```
}
```

C++

```
void OnOLECompleteDrag(long Effect)
{
}
```

C++ Builder

```
void __fastcall OLECompleteDrag(TObject *Sender,long Effect)
{
}
```

Delphi

```
procedure OLECompleteDrag(ASender: TObject; Effect : Integer);
begin
end;
```

Delphi 8 (.NET only)

```
procedure OLECompleteDrag(sender: System.Object; e:
AxEXTHUMBNAILLib._IThumbnailEvents_OLECompleteDragEvent);
begin
end;
```

Powe…

```
begin event OLECompleteDrag(long Effect)
end event OLECompleteDrag
```

VB.NET

```
Private Sub OLECompleteDrag(ByVal sender As System.Object, ByVal e As
AxEXTHUMBNAILLib._IThumbnailEvents_OLECompleteDragEvent) Handles
OLECompleteDrag
End Sub
```

VB6

```
Private Sub OLECompleteDrag(ByVal Effect As Long)
End Sub
```

VBA

```
Private Sub OLECompleteDrag(ByVal Effect As Long)
End Sub
```

VFP

```
LPARAMETERS Effect
```

Xbas…

```
PROCEDURE OnOLECompleteDrag(oThumbnail,Effect)
RETURN
```

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

Java…

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

VBSc…

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

Visual Data…

```
Procedure OnComOLECompleteDrag Integer llEffect
   Forward Send OnComOLECompleteDrag llEffect
End_Procedure
```

Visual Objects

```
METHOD OCX_OLECompleteDrag(Effect) CLASS MainDialog
RETURN NIL
```

X++

```
// OLECompleteDrag event is not supported. Use the
DragEnter,DragLeave,DragOver, DragDrop ... events.
```

XBasic

```
function OLECompleteDrag as v (Effect as N)
end function
```

dBASE

```
function nativeObject_OLECompleteDrag(Effect)
return
```

event OLEDragDrop (Data as ExDataObject, Effect as Long, Button as Integer, Shift as Integer, X as OLE_XPOS_PIXELS, Y as OLE_YPOS_PIXELS)

Occurs when a source component is dropped onto a target component when the source component determines that a drop can occur.

Type	Description
Data as ExDataObject	An ExDataObject object containing formats that the source will provide and, in addition, possibly the data for those formats. If no data is contained in the ExDataObject, it is provided when the control calls the GetData method. The SetData and Clear methods cannot be used here.
Effect as Long	A Long set by the target component identifying the action that has been performed (if any), thus allowing the source to take appropriate action if the component was moved (such as the source deleting the data). The possible values are listed in Remarks.
Button as Integer	An integer which acts as a bit field corresponding to the state of a mouse button when it is depressed. The left button is bit 0, the right button is bit 1, and the middle button is bit 2. These bits correspond to the values 1, 2, and 4, respectively. It indicates the state of the mouse buttons; some, all, or none of these three bits can be set, indicating that some, all, or none of the buttons are depressed.
Shift as Integer	An integer which acts as a bit field corresponding to the state of the SHIFT, CTRL, and ALT keys when they are depressed. The SHIFT key is bit 0, the CTRL key is bit 1, and the ALT key is bit 2. These bits correspond to the values 1, 2, and 4, respectively. The shift parameter indicates the state of these keys; some, all, or none of the bits can be set, indicating that some, all, or none of the keys are depressed. For example, if both the CTRL and ALT keys were depressed, the value of shift would be 6.
X as OLE_XPOS_PIXELS	A single that specifies the current X location of the mouse pointer. The X value is always expressed in container coordinates
Y as OLE_YPOS_PIXELS	A single that specifies the current Y location of the mouse pointer. The Y value is always expressed in container coordinates.

In the /NET Assembly, you have to use the DragDrop event as explained here:

- *https://www.exontrol.com/sg.jsp?content=support/faq/net/#dragdrop*

In the /COM version, you have to set the OLEDropMode property as explained here:

- *https://www.exontrol.com/sg.jsp?content=support/faq/all/#dragdrop*

The OLEDragDrop event is fired when the user has dropped files or clipboard information into the control. Use the OLEDropMode property on exOLEDropManual to enable OLE drop and drop support.

The settings for Effect are:

- exOLEDropEffectNone (0), Drop target cannot accept the data, or the drop operation was cancelled
- exOLEDropEffectCopy (1), Drop results in a copy of data from the source to the target. The original data is unaltered by the drag operation.
- exOLEDropEffectMove (2), Drop results in data being moved from drag source to drop source. The drag source should remove the data from itself after the move.

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

C#
```
// OLEDragDrop event is not supported. Use the DragEnter,DragLeave,DragOver,
DragDrop ... events.
```

VB
```
// OLEDragDrop event is not supported. Use the DragEnter,DragLeave,DragOver,
DragDrop ... events.
```

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

C#
```
private void OLEDragDrop(object sender,
AxEXTHUMBNAILLib._IThumbnailEvents_OLEDragDropEvent e)
{
}
```

C++
```
void OnOLEDragDrop(LPDISPATCH Data,long FAR* Effect,short Button,short
Shift,long X,long Y)
{
}
```

C++ Builder

```
void __fastcall OLEDragDrop(TObject *Sender,Exthumbnaillib_tlb::IExDataObject
*Data,long * Effect,short Button,short Shift,int X,int Y)
{
}
```

Delphi

```
procedure OLEDragDrop(ASender: TObject; Data : IExDataObject;var Effect :
Integer;Button : Smallint;Shift : Smallint;X : Integer;Y : Integer);
begin
end;
```

Delphi 8 (.NET only)

```
procedure OLEDragDrop(sender: System.Object; e:
AxEXTHUMBNAILLib._IThumbnailEvents_OLEDragDropEvent);
begin
end;
```

Powe…

```
begin event OLEDragDrop(oleobject Data,long Effect,integer Button,integer
Shift,long X,long Y)
end event OLEDragDrop
```

VB.NET

```
Private Sub OLEDragDrop(ByVal sender As System.Object, ByVal e As
AxEXTHUMBNAILLib._IThumbnailEvents_OLEDragDropEvent) Handles
OLEDragDrop
End Sub
```

VB6

```
Private Sub OLEDragDrop(ByVal Data As
EXTHUMBNAILLibCtl.IExDataObject,Effect As Long,ByVal Button As Integer,ByVal
Shift As Integer,ByVal X As Single,ByVal Y As Single)
End Sub
```

VBA

```
Private Sub OLEDragDrop(ByVal Data As Object,Effect As Long,ByVal Button As
Integer,ByVal Shift As Integer,ByVal X As Long,ByVal Y As Long)
End Sub
```

VFP

```
LPARAMETERS Data,Effect,Button,Shift,X,Y
```

Xbas…

```
PROCEDURE OnOLEDragDrop(oThumbnail,Data,Effect,Button,Shift,X,Y)
RETURN
```

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

Java…

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

VBSc…

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

Visual Data…

```
Procedure OnComOLEDragDrop Variant llData Integer llEffect Short llButton
Short llShift OLE_XPOS_PIXELS llX OLE_YPOS_PIXELS llY
   Forward Send OnComOLEDragDrop llData llEffect llButton llShift llX llY
End_Procedure
```

Visual Objects

```
METHOD OCX_OLEDragDrop(Data,Effect,Button,Shift,X,Y) CLASS MainDialog
RETURN NIL
```

X++

```
// OLEDragDrop event is not supported. Use the DragEnter,DragLeave,DragOver,
DragDrop ... events.
```

XBasic

```
function OLEDragDrop as v (Data as
OLE::Exontrol.Thumbnail.1::IExDataObject,Effect as N,Button as N,Shift as N,X as
OLE::Exontrol.Thumbnail.1::OLE_XPOS_PIXELS,Y as
OLE::Exontrol.Thumbnail.1::OLE_YPOS_PIXELS)
end function
```

dBASE

```
function nativeObject_OLEDragDrop(Data,Effect,Button,Shift,X,Y)
return
```

event OLEDragOver (Data as ExDataObject, Effect as Long, Button as Integer, Shift as Integer, X as OLE_XPOS_PIXELS, Y as OLE_YPOS_PIXELS, State as Integer)

Occurs when one component is dragged over another.

Type	Description
Data as ExDataObject	An ExDataObject object containing formats that the source will provide and, in addition, possibly the data for those formats. If no data is contained in the ExDataObject, it is provided when the control calls the GetData method. The SetData and Clear methods cannot be used here
Effect as Long	A Long set by the target component identifying the action that has been performed (if any), thus allowing the source to take appropriate action if the component was moved (such as the source deleting the data). The possible values are listed in Remarks.
Button as Integer	An integer which acts as a bit field corresponding to the state of a mouse button when it is depressed. The left button is bit 0, the right button is bit 1, and the middle button is bit 2. These bits correspond to the values 1, 2, and 4, respectively. It indicates the state of the mouse buttons; some, all, or none of these three bits can be set, indicating that some, all, or none of the buttons are depressed.
Shift as Integer	An integer which acts as a bit field corresponding to the state of the SHIFT, CTRL, and ALT keys when they are depressed. The SHIFT key is bit 0, the CTRL key is bit 1, and the ALT key is bit 2. These bits correspond to the values 1, 2, and 4, respectively. The shift parameter indicates the state of these keys; some, all, or none of the bits can be set, indicating that some, all, or none of the keys are depressed. For example, if both the CTRL and ALT keys were depressed, the value of shift would be 6.
X as OLE_XPOS_PIXELS	A single that specifies the current X location of the mouse pointer. The X value is always expressed in container coordinates.
Y as OLE_YPOS_PIXELS	A single that specifies the current Y location of the mouse pointer. The Y value is always expressed in container coordinates.

State as Integer	An integer that corresponds to the transition state of the control being dragged in relation to a target form or control. The possible values are listed in Remarks.

The settings for effect are:

- exOLEDropEffectNone (0), Drop target cannot accept the data, or the drop operation was cancelled
- exOLEDropEffectCopy (1), Drop results in a copy of data from the source to the target. The original data is unaltered by the drag operation.
- exOLEDropEffectMove (2), Drop results in data being moved from drag source to drop source. The drag source should remove the data from itself after the move.

The settings for state are:

- exOLEDragEnter (0), Source component is being dragged within the range of a target.
- exOLEDragLeave (1), Source component is being dragged out of the range of a target.
- exOLEOLEDragOver (2), Source component has moved from one position in the target to another.

Note If the state parameter is 1, indicating that the mouse pointer has left the target, then the x and y parameters will contain zeros.
The source component should always mask values from the effect parameter to ensure compatibility with future implementations of ActiveX components. As a precaution against future problems, drag sources and drop targets should mask these values appropriately before performing any comparisons.
For example, a source component should not compare an effect against, say, exOLEDropEffectCopy, such as in this manner:

If Effect = exOLEDropEffectCopy...
Instead, the source component should mask for the value or values being sought, such as this:
If Effect And exOLEDropEffectCopy = exOLEDropEffectCopy...
-or-
If (Effect And exOLEDropEffectCopy)...
This allows for the definition of new drop effects in future versions while preserving backwards compatibility with your existing code.
The control supports only manual OLE drag and drop events.

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

C#
```
// OLEDragOver event is not supported. Use the DragEnter,DragLeave,DragOver,
DragDrop ... events.
```

VB

```
// OLEDragOver event is not supported. Use the DragEnter,DragLeave,DragOver,
DragDrop ... events.
```

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

C#

```
private void OLEDragOver(object sender,
AxEXTHUMBNAILLib._IThumbnailEvents_OLEDragOverEvent e)
{
}
```

C++

```
void OnOLEDragOver(LPDISPATCH Data,long FAR* Effect,short Button,short
Shift,long X,long Y,short State)
{
}
```

C++ Builder

```
void __fastcall OLEDragOver(TObject *Sender,Exthumbnaillib_tlb::IExDataObject
*Data,long * Effect,short Button,short Shift,int X,int Y,short State)
{
}
```

Delphi

```
procedure OLEDragOver(ASender: TObject; Data : IExDataObject;var Effect :
Integer;Button : Smallint;Shift : Smallint;X : Integer;Y : Integer;State : Smallint);
begin
end;
```

Delphi 8 (.NET only)

```
procedure OLEDragOver(sender: System.Object; e:
AxEXTHUMBNAILLib._IThumbnailEvents_OLEDragOverEvent);
begin
end;
```

Powe…

```
begin event OLEDragOver(oleobject Data,long Effect,integer Button,integer
Shift,long X,long Y,integer State)
end event OLEDragOver
```

VB.NET

```
Private Sub OLEDragOver(ByVal sender As System.Object, ByVal e As
AxEXTHUMBNAILLib._IThumbnailEvents_OLEDragOverEvent) Handles
OLEDragOver
End Sub
```

VB6

```
Private Sub OLEDragOver(ByVal Data As EXTHUMBNAILLibCtl.IExDataObject,Effect As Long,ByVal Button As Integer,ByVal Shift As Integer,ByVal X As Single,ByVal Y As Single,ByVal State As Integer)
End Sub
```

VBA

```
Private Sub OLEDragOver(ByVal Data As Object,Effect As Long,ByVal Button As Integer,ByVal Shift As Integer,ByVal X As Long,ByVal Y As Long,ByVal State As Integer)
End Sub
```

VFP

```
LPARAMETERS Data,Effect,Button,Shift,X,Y,State
```

Xbas…

```
PROCEDURE OnOLEDragOver(oThumbnail,Data,Effect,Button,Shift,X,Y,State)
RETURN
```

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

Java…

```
<SCRIPT EVENT="OLEDragOver(Data,Effect,Button,Shift,X,Y,State)"
LANGUAGE="JScript">
</SCRIPT>
```

VBSc…

```
<SCRIPT LANGUAGE="VBScript">
Function OLEDragOver(Data,Effect,Button,Shift,X,Y,State)
End Function
</SCRIPT>
```

Visual Data…

```
Procedure OnComOLEDragOver Variant llData Integer llEffect Short llButton Short llShift OLE_XPOS_PIXELS llX OLE_YPOS_PIXELS llY Short llState
    Forward Send OnComOLEDragOver llData llEffect llButton llShift llX llY llState
End_Procedure
```

Visual Objects

```
METHOD OCX_OLEDragOver(Data,Effect,Button,Shift,X,Y,State) CLASS MainDialog
RETURN NIL
```

X++

```
// OLEDragOver event is not supported. Use the DragEnter,DragLeave,DragOver, DragDrop ... events.
```

XBasic

```
function OLEDragOver as v (Data as
OLE::Exontrol.Thumbnail.1::IExDataObject,Effect as N,Button as N,Shift as N,X as
OLE::Exontrol.Thumbnail.1::OLE_XPOS_PIXELS,Y as
OLE::Exontrol.Thumbnail.1::OLE_YPOS_PIXELS,State as N)
end function
```

dBASE

```
function nativeObject_OLEDragOver(Data,Effect,Button,Shift,X,Y,State)
return
```

event OLEGiveFeedback (Effect as Long, DefaultCursors as Boolean)

Allows the drag source to specify the type of OLE drag-and-drop operation and the visual feedback.

Type	Description
Effect as Long	A long integer set by the target component in the OLEDragOver event specifying the action to be performed if the user drops the selection on it. This allows the source to take the appropriate action (such as giving visual feedback). The possible values are listed in Remarks.
DefaultCursors as Boolean	Boolean value that determines whether to use the default mouse cursor, or to use a user-defined mouse cursor.True (default) = use default mouse cursor.False = do not use default cursor. Mouse cursor must be set with the MousePointer property of the Screen object.

The settings for Effect are:

- exOLEDropEffectNone (0), Drop target cannot accept the data, or the drop operation was cancelled
- exOLEDropEffectCopy (1), Drop results in a copy of data from the source to the target. The original data is unaltered by the drag operation.
- exOLEDropEffectMove (2), Drop results in data being moved from drag source to drop source. The drag source should remove the data from itself after the move.

If there is no code in the OLEGiveFeedback event, or if the defaultcursors parameter is set to True, the mouse cursor will be set to the default cursor provided by the control. The source component should always mask values from the effect parameter to ensure compatibility with future implementations of ActiveX components. As a precaution against future problems, drag sources and drop targets should mask these values appropriately before performing any comparisons.

For example, a source component should not compare an effect against, say, exOLEDropEffectCopy, such as in this manner:
If Effect = exOLEDropEffectCopy...
Instead, the source component should mask for the value or values being sought, such as this:
If Effect And exOLEDropEffectCopy = exOLEDropEffectCopy...
-or-
If (Effect And exOLEDropEffectCopy)...
This allows for the definition of new drop effects in future versions while preserving backwards compatibility with your existing code.

The control supports only manual OLE drag and drop events.

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

C#
```
// OLEGiveFeedback event is not supported. Use the
DragEnter,DragLeave,DragOver, DragDrop ... events.
```

VB
```
// OLEGiveFeedback event is not supported. Use the
DragEnter,DragLeave,DragOver, DragDrop ... events.
```

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

C#
```
private void OLEGiveFeedback(object sender,
AxEXTHUMBNAILLib._IThumbnailEvents_OLEGiveFeedbackEvent e)
{
}
```

C++
```
void OnOLEGiveFeedback(long Effect,BOOL FAR* DefaultCursors)
{
}
```

C++ Builder
```
void __fastcall OLEGiveFeedback(TObject *Sender,long Effect,VARIANT_BOOL *
DefaultCursors)
{
}
```

Delphi
```
procedure OLEGiveFeedback(ASender: TObject; Effect : Integer;var DefaultCursors
: WordBool);
begin
end;
```

Delphi 8 (.NET only)
```
procedure OLEGiveFeedback(sender: System.Object; e:
AxEXTHUMBNAILLib._IThumbnailEvents_OLEGiveFeedbackEvent);
begin
end;
```

Powe…
```
begin event OLEGiveFeedback(long Effect,boolean DefaultCursors)
end event OLEGiveFeedback
```

VB.NET

```
Private Sub OLEGiveFeedback(ByVal sender As System.Object, ByVal e As
AxEXTHUMBNAILLib._IThumbnailEvents_OLEGiveFeedbackEvent) Handles
OLEGiveFeedback
End Sub
```

VB6

```
Private Sub OLEGiveFeedback(ByVal Effect As Long,DefaultCursors As Boolean)
End Sub
```

VBA

```
Private Sub OLEGiveFeedback(ByVal Effect As Long,DefaultCursors As Boolean)
End Sub
```

VFP

```
LPARAMETERS Effect,DefaultCursors
```

Xbas…

```
PROCEDURE OnOLEGiveFeedback(oThumbnail,Effect,DefaultCursors)
RETURN
```

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

Java…

```
<SCRIPT EVENT="OLEGiveFeedback(Effect,DefaultCursors)"
LANGUAGE="JScript">
</SCRIPT>
```

VBSc…

```
<SCRIPT LANGUAGE="VBScript">
Function OLEGiveFeedback(Effect,DefaultCursors)
End Function
</SCRIPT>
```

Visual Data…

```
Procedure OnComOLEGiveFeedback Integer llEffect Boolean llDefaultCursors
    Forward Send OnComOLEGiveFeedback llEffect llDefaultCursors
End_Procedure
```

Visual Objects

```
METHOD OCX_OLEGiveFeedback(Effect,DefaultCursors) CLASS MainDialog
RETURN NIL
```

X++

```
// OLEGiveFeedback event is not supported. Use the
DragEnter,DragLeave,DragOver, DragDrop ... events.
```

XBasic

```
function OLEGiveFeedback as v (Effect as N,DefaultCursors as L)
end function
```

dBASE

```
function nativeObject_OLEGiveFeedback(Effect,DefaultCursors)
return
```

event OLESetData (Data as ExDataObject, Format as Integer)

Occurs on a drag source when a drop target calls the GetData method and there is no data in a specified format in the OLE drag-and-drop DataObject.

Type	Description
Data as ExDataObject	An ExDataObject object in which to place the requested data. The component calls the SetData method to load the requested format.
Format as Integer	An integer specifying the format of the data that the target component is requesting. The source component uses this value to determine what to load into the ExDataObject object.

The OLESetData is not currently supported.

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

C#
```
// OLESetData event is not supported. Use the DragEnter,DragLeave,DragOver, DragDrop ... events.
```

VB
```
// OLESetData event is not supported. Use the DragEnter,DragLeave,DragOver, DragDrop ... events.
```

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

C#
```
private void OLESetData(object sender, AxEXTHUMBNAILLib._IThumbnailEvents_OLESetDataEvent e)
{
}
```

C++
```
void OnOLESetData(LPDISPATCH Data,short Format)
{
}
```

C++ Builder
```
void __fastcall OLESetData(TObject *Sender,Exthumbnaillib_tlb::IExDataObject *Data,short Format)
{
}
```

Delphi

```
procedure OLESetData(ASender: TObject; Data : IExDataObject;Format : Smallint);
begin
end;
```

Delphi 8 (.NET only)

```
procedure OLESetData(sender: System.Object; e:
AxEXTHUMBNAILLib._IThumbnailEvents_OLESetDataEvent);
begin
end;
```

Powe…

```
begin event OLESetData(oleobject Data,integer Format)
end event OLESetData
```

VB.NET

```
Private Sub OLESetData(ByVal sender As System.Object, ByVal e As
AxEXTHUMBNAILLib._IThumbnailEvents_OLESetDataEvent) Handles OLESetData
End Sub
```

VB6

```
Private Sub OLESetData(ByVal Data As EXTHUMBNAILLibCtl.IExDataObject,ByVal
Format As Integer)
End Sub
```

VBA

```
Private Sub OLESetData(ByVal Data As Object,ByVal Format As Integer)
End Sub
```

VFP

```
LPARAMETERS Data,Format
```

Xbas…

```
PROCEDURE OnOLESetData(oThumbnail,Data,Format)
RETURN
```

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

Java…

```
<SCRIPT EVENT="OLESetData(Data,Format)" LANGUAGE="JScript">
</SCRIPT>
```

VBSc…

```
<SCRIPT LANGUAGE="VBScript">
Function OLESetData(Data,Format)
End Function
```

```
</SCRIPT>
```

Visual Data…

```
Procedure OnComOLESetData Variant llData Short llFormat
    Forward Send OnComOLESetData llData llFormat
End_Procedure
```

Visual Objects

```
METHOD OCX_OLESetData(Data,Format) CLASS MainDialog
RETURN NIL
```

X++

```
// OLESetData event is not supported. Use the DragEnter,DragLeave,DragOver,
DragDrop ... events.
```

XBasic

```
function OLESetData as v (Data as
OLE::Exontrol.Thumbnail.1::IExDataObject,Format as N)
end function
```

dBASE

```
function nativeObject_OLESetData(Data,Format)
return
```

event OLEStartDrag (Data as ExDataObject, AllowedEffects as Long)

Occurs when the OLEDrag method is called.

Type	Description
Data as ExDataObject	An ExDataObject object containing formats that the source will provide and, optionally, the data for those formats. If no data is contained in the ExDataObject, it is provided when the control calls the GetData method. The programmer should provide the values for this parameter in this event. The SetData and Clear methods cannot be used here.
AllowedEffects as Long	A long containing the effects that the source component supports. The possible values are listed in Settings. The programmer should provide the values for this parameter in this event

In the /NET Assembly, you have to use the DragEnter event as explained here:

- *https://www.exontrol.com/sg.jsp?content=support/faq/net/#dragdrop*

For /COM components, you can follow the next tutorial in order to implement the OLE Drag and Drop:

- https://www.exontrol.com/sg.jsp?content=support/faq/all/#dragdrop

The settings for AllowEffects are:

- exOLEDropEffectNone (0), Drop target cannot accept the data, or the drop operation was cancelled
- exOLEDropEffectCopy (1), Drop results in a copy of data from the source to the target. The original data is unaltered by the drag operation.
- exOLEDropEffectMove (2), Drop results in data being moved from drag source to drop source. The drag source should remove the data from itself after the move.

The source component should logically Or together the supported values and places the result in the AllowedEffects parameter. The target component can use this value to determine the appropriate action (and what the appropriate user feedback should be). You may wish to defer putting data into the ExDataObject object until the target component requests it. This allows the source component to save time. If the user does not load any formats into the ExDataObject, then the drag/drop operation is canceled. Use exCFFiles and Files property to add files to the drag and drop data object.

The idea of drag and drop in exThumbnail control is the same as in other controls. To start

accepting drag and drop sources the exThumbnail control should have the OLEDropMode to exOLEDropManual. Once that is is set, the exThumbnail starts accepting any drag and drop sources.

The first step is if you want to be able to drag items from your exThumbnail control to other controls the idea is to handle the OLE_StartDrag event. The event passes an object ExDataObject (Data) as argument. The Data and AllowedEffects can be changed only in the OLEStartDrag event. The OLE_StartDrag event is fired when user is about to drag items from the control.

The AllowedEffect parameter and SetData property must be set to continue a drag and drop operation.

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

C#
```
// OLEStartDrag event is not supported. Use the DragEnter,DragLeave,DragOver,
DragDrop ... events.
```

VB
```
// OLEStartDrag event is not supported. Use the DragEnter,DragLeave,DragOver,
DragDrop ... events.
```

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

C#
```
private void OLEStartDrag(object sender,
AxEXTHUMBNAILLib._IThumbnailEvents_OLEStartDragEvent e)
{
}
```

C++
```
void OnOLEStartDrag(LPDISPATCH Data,long FAR* AllowedEffects)
{
}
```

C++ Builder
```
void __fastcall OLEStartDrag(TObject *Sender,Exthumbnaillib_tlb::IExDataObject
*Data,long * AllowedEffects)
{
}
```

Delphi
```
procedure OLEStartDrag(ASender: TObject; Data : IExDataObject;var
AllowedEffects : Integer);
begin
end;
```

Delphi 8 (.NET only)

```
procedure OLEStartDrag(sender: System.Object; e:
AxEXTHUMBNAILLib._IThumbnailEvents_OLEStartDragEvent);
begin
end;
```

Powe…

```
begin event OLEStartDrag(oleobject Data,long AllowedEffects)
end event OLEStartDrag
```

VB.NET

```
Private Sub OLEStartDrag(ByVal sender As System.Object, ByVal e As
AxEXTHUMBNAILLib._IThumbnailEvents_OLEStartDragEvent) Handles
OLEStartDrag
End Sub
```

VB6

```
Private Sub OLEStartDrag(ByVal Data As
EXTHUMBNAILLibCtl.IExDataObject,AllowedEffects As Long)
End Sub
```

VBA

```
Private Sub OLEStartDrag(ByVal Data As Object,AllowedEffects As Long)
End Sub
```

VFP

```
LPARAMETERS Data,AllowedEffects
```

Xbas…

```
PROCEDURE OnOLEStartDrag(oThumbnail,Data,AllowedEffects)
RETURN
```

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

Java…

```
<SCRIPT EVENT="OLEStartDrag(Data,AllowedEffects)" LANGUAGE="JScript">
</SCRIPT>
```

VBSc…

```
<SCRIPT LANGUAGE="VBScript">
Function OLEStartDrag(Data,AllowedEffects)
End Function
</SCRIPT>
```

Visual Data…

```
Procedure OnComOLEStartDrag Variant llData Integer llAllowedEffects
   Forward Send OnComOLEStartDrag llData llAllowedEffects
End_Procedure
```

Visual Objects

```
METHOD OCX_OLEStartDrag(Data,AllowedEffects) CLASS MainDialog
RETURN NIL
```

X++

```
// OLEStartDrag event is not supported. Use the DragEnter,DragLeave,DragOver,
DragDrop ... events.
```

XBasic

```
function OLEStartDrag as v (Data as
OLE::Exontrol.Thumbnail.1::IExDataObject,AllowedEffects as N)
end function
```

dBASE

```
function nativeObject_OLEStartDrag(Data,AllowedEffects)
return
```

event StateChange (State as StateChangeEnum)

Fired while the control's state has been changed.

Type	Description
State as StateChangeEnum	A StateChangeEnum expression that indicates the control's state.

The StateChange event occurs once the control's state is changed. For instance, you can use the StateChange event to handle selection change within the control. Also, you can use the StateChange event to respond to any context menu operations. The ShowContextMenu property indicates the items to be displayed on the object's context menu.

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

C#

```
private void StateChange(object
sender,exontrol.EXTHUMBNAILLib.StateChangeEnum   State)
{
}
```

VB

```
Private Sub StateChange(ByVal sender As System.Object,ByVal State As
exontrol.EXTHUMBNAILLib.StateChangeEnum) Handles StateChange
End Sub
```

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

C#

```
private void StateChange(object sender,
AxEXTHUMBNAILLib._IThumbnailEvents_StateChangeEvent e)
{
}
```

C++

```
void OnStateChange(long   State)
{
}
```

C++ Builder

```
void __fastcall StateChange(TObject *Sender,Exthumbnaillib_tlb::StateChangeEnum
State)
{
}
```

Delphi

```
procedure StateChange(ASender: TObject; State : StateChangeEnum);
begin
end;
```

Delphi 8 (.NET only)

```
procedure StateChange(sender: System.Object; e:
AxEXTHUMBNAILLib._IThumbnailEvents_StateChangeEvent);
begin
end;
```

Powe…

```
begin event StateChange(long  State)

end event StateChange
```

VB.NET

```
Private Sub StateChange(ByVal sender As System.Object, ByVal e As
AxEXTHUMBNAILLib._IThumbnailEvents_StateChangeEvent) Handles StateChange
End Sub
```

VB6

```
Private Sub StateChange(ByVal State As EXTHUMBNAILLibCtl.StateChangeEnum)
End Sub
```

VBA

```
Private Sub StateChange(ByVal State As Long)
End Sub
```

VFP

```
LPARAMETERS State
```

Xbas…

```
PROCEDURE OnStateChange(oThumbnail,State)

RETURN
```

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

Java…

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

VBSc…

```
<SCRIPT LANGUAGE="VBScript">
Function StateChange(State)
```

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

Visual Data…

```
Procedure OnComStateChange OLEStateChangeEnum   llState
   Forward Send OnComStateChange llState
End_Procedure
```

Visual Objects

```
METHOD OCX_StateChange(State) CLASS MainDialog
RETURN NIL
```

X++

```
void onEvent_StateChange(int   _State)
{
}
```

XBasic

```
function StateChange as v (State  as
OLE::Exontrol.Thumbnail.1::StateChangeEnum)
end function
```

dBASE

```
function nativeObject_StateChange(State)
return
```

Expressions

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

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

Usage examples:

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

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

The constants can be represented as:

- numbers in **decimal** format (where dot character specifies the decimal separator). For instance: -1, 100, 20.45, .99 and so on
- numbers in **hexa-decimal** format (preceded by **0x** or **0X** sequence), uses sixteen distinct symbols, most often the symbols 0-9 to represent values zero to nine, and A, B, C, D, E, F (or alternatively a, b, c, d, e, f) to represent values ten to fifteen. Hexadecimal numerals are widely used by computer system designers and programmers. As each hexadecimal digit represents four binary digits (bits), it allows a more human-friendly representation of binary-coded values. For instance, 0xFF,

0x00FF00, and so so.

- **date-time** in format **#mm/dd/yyyy hh:mm:ss#**, For instance, #1/31/2001 10:00# means the January 31th, 2001, 10:00 AM
- **string**, if it starts / ends with any of the **'** or ` or **"** characters. If you require the starting character inside the string, it should be escaped ( preceded by a \ character ). For instance, `Mihai`, "Filimon", 'has', "\"a quote\"", and so on

The predefined constants are:

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

The supported binary arithmetic operators are:

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

The supported unary boolean operators are:

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

The supported binary boolean operators are:

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

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

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

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

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

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

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

  variable := expression

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

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

  =: variable

  where variable is a integer between 0 and 9. You can use the **:=** operator to store the value of any expression (please make the difference between := and =:). For

instance, *(0:=dbl(value)) = 0 ? "zero" : =:0*, stores the value converted to double, and prints zero if it is 0, else the converted number. Please pay attention that the **:=** and **=:** are two distinct operators, the first for storing the result into a variable, while the second for restoring the variable

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

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

  expression ? true_part : false_part

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

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

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

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

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

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

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

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

statement is used, it could take up to 1000 operations to check, so by far, the *in* operator could save time on finding elements within a collection.

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

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

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

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

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

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

  - #4/1/2009#, from hours 06:00 AM to 12:00 PM
  - #4/5/2009#, from hours 07:00 AM to 10:00 AM and hours 03:00PM, 04:00PM, 06:00PM and 10:00PM
  - #5/1/2009#, from hours 12:00 AM to 08:00 AM

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

The supported conversion unary operators are:

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

The bitwise operators for numbers are:

- a **bitand** b (binary operator) computes the AND operation on bits of a and b, and returns the unsigned value. For instance, 0x01001000 bitand 0x10111000 returns 0x00001000.
- a **bitor** b (binary operator) computes the OR operation on bits of a and b, and returns the unsigned value. For instance, 0x01001000 bitor 0x10111000 returns 0x11111000.
- a **bitxor** b (binary operator) computes the XOR (exclusive-OR) operation on bits of a and b, and returns the unsigned value. For instance, 0x01110010 bitxor 0x10101010 returns 0x11011000.
- a **bitshift** (b) (binary operator) shifts every bit of a value to the left if b is negative, or to the right if b is positive, for b times, and returns the unsigned value. For instance, 128 bitshift 1 returns 64 (dividing by 2) or 128 bitshift (-1) returns 256 (multiplying by

2)

- **bitnot** (unary operator) flips every bit of x, and returns the unsigned value. For instance, bitnot(0x00FF0000) returns 0xFF00FFFF.

The operators for numbers are:

- **int** (unary operator) retrieves the integer part of the number. For instance, the *int(12.54)* returns 12
- **round** (unary operator) rounds the number ie 1.2 gets 1, since 1.8 gets 2. For instance, the *round(12.54)* returns 13
- **floor** (unary operator) returns the largest number with no fraction part that is not greater than the value of its argument. For instance, the *floor(12.54)* returns 12
- **abs** (unary operator) retrieves the absolute part of the number ie -1 gets 1, 2 gets 2. For instance, the *abs(-12.54)* returns 12.54
- **sin** (unary operator) returns the sine of an angle of x radians. For instance, the *sin(3.14)* returns 0.001593.
- **cos** (unary operator) returns the cosine of an angle of x radians. For instance, the *cos(3.14)* returns -0.999999.
- **asin** (unary operator) returns the principal value of the arc sine of x, expressed in radians. For instance, the *2*asin(1)* returns the value of PI.
- **acos** (unary operator) returns the principal value of the arc cosine of x, expressed in radians. For instance, the *2*acos(0)* returns the value of PI
- **sqrt** (unary operator) returns the square root of x. For instance, the *sqrt(81)* returns 9.
- **currency** (unary operator) formats the giving number as a currency string, as indicated by the control panel. For instance, *currency(value)* displays the value using the current format for the currency ie, 1000 gets displayed as $1,000.00, for US format.
- value **format** 'flags' (binary operator) formats the value with specified flags. If flags is empty, the number is displayed as shown in the field "Number" in the "Regional and Language Options" from the Control Panel. For instance the *1000 format ''* displays 1,000.00 for English format, while 1.000,00 is displayed for German format. 1000 format '2|.|3|,' will always displays 1,000.00 no matter of settings in the control panel. If formatting the number fails for some invalid parameter, the value is displayed with no formatting.

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

  - *NumDigits* - specifies the number of fractional digits, If the flag is missing, the field "No. of digits after decimal" from "Regional and Language Options" is using.
  - *DecimalSep* - specifies the decimal separator. If the flag is missing, the field "Decimal symbol" from "Regional and Language Options" is using.
  - Grouping - indicates the number of digits in each group of numbers to the left of

the decimal separator. Values in the range 0 through 9 and 32 are valid. The most significant grouping digit indicates the number of digits in the least significant group immediately to the left of the decimal separator. Each subsequent grouping digit indicates the next significant group of digits to the left of the previous group. If the last value supplied is not 0, the remaining groups repeat the last group. Typical examples of settings for this member are: 0 to group digits as in 123456789.00; 3 to group digits as in 123,456,789.00; and 32 to group digits as in 12,34,56,789.00. If the flag is missing, the field "Digit grouping" from "Regional and Language Options" indicates the grouping flag.

- *ThousandSep* - specifies the thousand separator. If the flag is missing, the field "Digit grouping symbol" from "Regional and Language Options" is using.
- *NegativeOrder* - indicates the negative number mode. If the flag is missing, the field "Negative number format" from "Regional and Language Options" is using. The valid values are 0, 1, 2, 3 and 4 with the following meanings:
  - 0 - Left parenthesis, number, right parenthesis; for example, (1.1)
  - 1 - Negative sign, number; for example, -1.1
  - 2 - Negative sign, space, number; for example, - 1.1
  - 3 - Number, negative sign; for example, 1.1-
  - 4 - Number, space, negative sign; for example, 1.1 -
- *LeadingZero* - indicates if leading zeros should be used in decimal fields. If the flag is missing, the field "Display leading zeros" from "Regional and Language Options" is using. The valid values are 0, 1

The operators for strings are:

- **len** (unary operator) retrieves the number of characters in the string. For instance, the *len("Mihai")* returns 5.
- **lower** (unary operator) returns a string expression in lowercase letters. For instance, the *lower("MIHAI")* returns "mihai"
- **upper** (unary operator) returns a string expression in uppercase letters. For instance, the *upper("mihai")* returns "MIHAI"
- **proper** (unary operator) returns from a character expression a string capitalized as appropriate for proper names. For instance, the *proper("mihai")* returns "Mihai"
- **ltrim** (unary operator) removes spaces on the left side of a string. For instance, the *ltrim(" mihai")* returns "mihai"
- **rtrim** (unary operator) removes spaces on the right side of a string. For instance, the *rtrim("mihai ")* returns "mihai"
- **trim** (unary operator) removes spaces on both sides of a string. For instance, the *trim(" mihai ")* returns "mihai"
- **reverse** (unary operator) reverses the order of the characters in the string a. For instance, the *reverse("Mihai")* returns "iahiM"
- a **startwith** b (binary operator) specifies whether a string starts with specified string (

0 if not found, -1 if found). For instance *"Mihai" startwith "Mi"* returns -1

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

The operators for dates are:

- **time** (unary operator) retrieves the time of the date in string format, as specified in the control's panel. For instance, the *time(#1/1/2001 13:00#)* returns "1:00:00 PM"

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

The expression supports also **immediate if** (similar with iif in visual basic, or ? : in C++) ie cond ? value_true : value_false, which means that once that cond is true the value_true is used, else the value_false is used. Also, it supports variables, up to 10 from 0 to 9. For instance, 0:="Abc" means that in the variable 0 is "Abc", and =:0 means retrieves the value of the variable 0. For instance, the len(%0) ? (0:=(%1+%2) ? currency(=:0) else `` ) : `` gets the sum between second and third column in currency format if it is not zero, and only if the first column is not empty. As you can see you can use the variables to avoid computing several times the same thing (in this case the sum %1 and %2 .