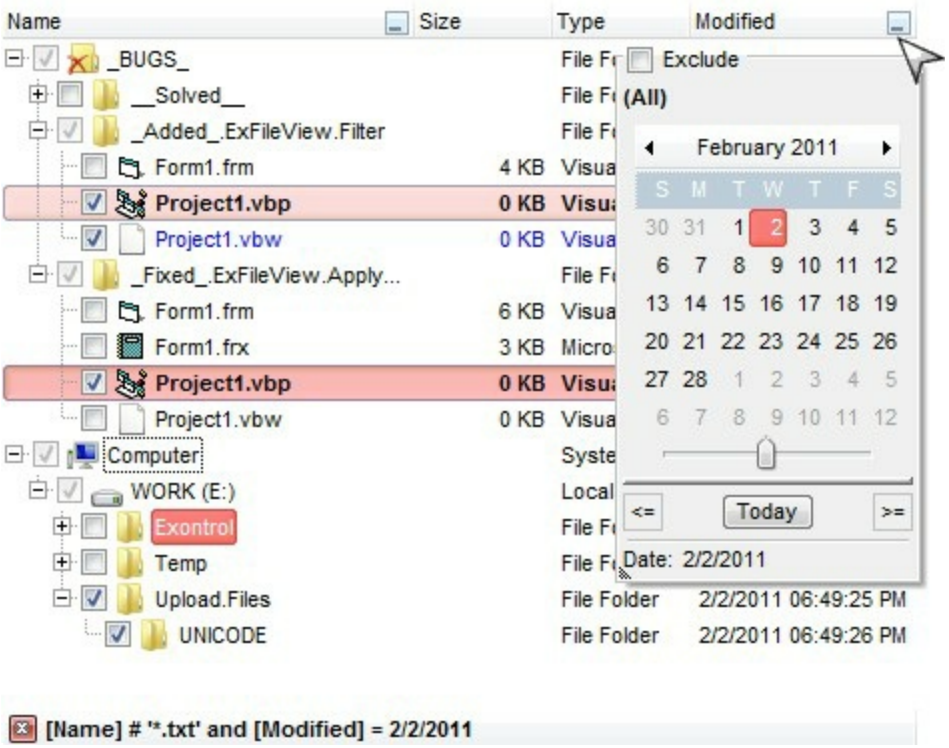
ExFileView

Provide rich display of file and folder information from within your applications. The ExFileView component allows creating Windows Explorer-style functionality. Files with different attributes can be displayed with different color, background color, font, etc. It can also filter the files based on files extensions using Include or Exclude clauses. The ExFileView component is able to change the displayed icon, or type file, supports Drag & Drop, incremental search, mouse wheel and more. The ExFileView control is able to highlight the folders that contains files changed into a given interval. The ExFileView control can simulate a FolderView control as well. Impress your customers with this powerful file/folder view control.

Features of the control include:

- **Print** and Print Preview support
- **OLE Drag and Drop** support
- **Skinnable Interface** support (ability to apply a skin to any background part)
- 'starts with' and 'contains' **incremental searching** support
- **CheckBox** support
- **Search** files and folders support
- Ability to filter files and folders on the fly, using the control's **filterbar**
- Files or folders with different attributes can be displayed with different color, background color, font, etc (conditional-format)
- Asynchronous support
- Ability to display the results from a recursive search
- Ability to filter files based on extensions using Include or Exclude clauses
- Ability to highlight the folders that contains files changed into a given interval
- Ability to change the displayed icon, or type file, for files or folders
- Ability to define custom filter patterns for any column
- Column sorting
- Mouse wheel support
- and more



How to get support?

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

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

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

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

Regards,
Exontrol Development Team

https://www.exontrol.com

constants AlignmentEnum

Specifies the object's alignment.

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

constants AppearanceEnum

The AppearanceEnum type specifies the control's appearance, as header appearance as well. Use the Appearance property to specify the control's appearance. Use the HeaderAppearance property to specify the header'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 a row or an item. The AutoDrag property indicates the way the component supports the AutoDrag feature. The AutoDrag feature indicates what the control does when the user clicks an item and start dragging. For instance, using the AutoDrag feature you can automatically lets the user to drag and drop the data to OLE compliant applications like Microsoft Word, Excel and so on.

- The flag that ends on **...OnShortTouch** indicates the action the control does when the user short touches the screen
- The flag that ends on **...OnRight** indicates the action the control does when the user right clicks the control.
- The flag that ends on **...OnLongTouch** indicates the action the control does when the user long touches the screen

The AutoDragEnum type supports the following values:

Name	Value	Description
exAutoDragNone	0	AutoDrag is disabled. You can use the OLEDropMode property to handle the OLE Drag and Drop event for your custom action.
exAutoDragCopy	8	Drag and drop the selected items to a target application, and paste them as image or text. Pasting the data to the target application depends on the application. You can use the exAutoDragCopyText to specify that you want to paste as Text, or exAutoDragCopyImage as an image.
exAutoDragCopyText	9	Drag and drop the selected items to a target application, and paste them as text only. Ability to drag and drop the data as text, to your favorite Office applications, like Word, Excel, or any other OLE-Automation compliant. The drag and drop operation can start anywhere Click here to watch a movie on how exAutoDragCopyText works.
		Drag and drop the selected items to a target application, and paste them as image only. Ability to drag and drop the data as it looks, to your favorite

exAutoDragCopyImage	10	Office applications, like Word, Excel, or any other OLE-Automation compliant. The drag and drop operation can start anywhere Click here to watch a movie on how exAutoDragCopyImage works.
exAutoDragCopySnapShot	11	Drag and drop a snap shot of the current component. This option could be used to drag and drop the current snap shot of the control to your favorite Office applications, like Word, Excel, or any other OLE-Automation compliant.
exAutoDragScroll	16	The component is scrolled by clicking the item and dragging to a new position. This option can be used to allow user scroll the control's content with NO usage of the scroll bar, like on your IPhone. Click here to watch a movie on how exAutoDragScroll works.
exAutoDragCopyOnShortTouch	2048	Drag and drop the selected objects to a target application, and paste them as image or text.
exAutoDragCopyTextOnShortTouch	2304	Drag and drop the selected objects to a target application, and paste them as text only.
exAutoDragCopyImageOnShortTouch	2560	Drag and drop the selected objects to a target application, and paste them as image only.
exAutoDragCopySnapShotOnShortTouch	2816	Drag and drop a snap shot of the current component.
exAutoDragScrollOnShortTouch	4096	The component is scrolled by clicking the object and dragging to a new position.
exAutoDragCopyOnRight	524288	Drag and drop the selected objects to a target application, and paste them as image or text.
exAutoDragCopyTextOnRight	589824	Drag and drop the selected objects to a target application, and paste them as text only.
exAutoDragCopyImageOnRight	655360	Drag and drop the selected objects to a target application, and paste them as image only.
exAutoDragCopySnapShotOnRight	720896	Drag and drop a snap shot of the current component.
exAutoDragScrollOnRight	1048576	The component is scrolled by clicking the object and

		dragging to a new position.
exAutoDragCopyOnLongTouch	134217728	Drag and drop the selected objects to a target application, and paste them as image or text.
exAutoDragCopyTextOnLongTouch	1561994944	Drag and drop the selected objects to a target application, and paste them as text only.
exAutoDragCopyImageOnLongTouch	1677721600	Drag and drop the selected objects to a target application, and paste them as image only.
exAutoDragCopySnapShotOnLongTouch	1845493678	Drag and drop a snap shot of the current component.
exAutoDragScrollOnLongTouch	268435456	The component is scrolled by clicking the object and dragging to a new position.

constants BackgroundPartEnum

The BackgroundPartEnum type indicates parts in the control. Use the Background property to specify a background color or a visual appearance for specific parts in the control. A Color expression that indicates the background color for a specified part. The last 7 bits in the high significant byte of the color to indicates the identifier of the skin being used. Use the Add method to add new skins to the control. If you need to remove the skin appearance from a part of the control you need to reset the last 7 bits in the high significant byte of the color being applied to the background's part. /*not supported in the lite version*/

If you refer a part of the scroll bar please notice the following:

- All BackgroundPartEnum expressions that starts with **exVS** changes a part in a vertical scroll bar
- All BackgroundPartEnum expressions that starts with **exHS** changes a part in the horizontal scroll bar
- Any BackgroundPartEnum expression that ends with **P** (and starts with exVS or exHS) specifies a part of the scrollbar when it is pressed.
- Any BackgroundPartEnum expression that ends with **D** (and starts with exVS or exHS) specifies a part of the scrollbar when it is disabled.
- Any BackgroundPartEnum expression that ends with **H** (and starts with exVS or exHS) specifies a part of the scrollbar when the cursor hovers it.
- Any BackgroundPartEnum expression that ends with no **H**, **P** or **D** (and starts with exVS or exHS) specifies a part of the scrollbar on normal state

Name	Value	Description
exHeaderFilterBarButton	0	Specifies the background color for the drop down filter bar button. Use the ColumnFilterButton property to specify whether the drop down filter bar button is visible or hidden.
exFooterFilterBarButton	1	Specifies the background color for the closing button in the filter bar. Use the ClearFilter method to remove the filter.
exShowFocusRect	19	Specifies the visual appearance to display the cell with the focus.
exSelBackColorFilter	20	exSelBackColorFilter. Specifies the visual appearance for the selection in the drop down filter window.
exSelForeColorFilter	21	exSelForeColorFilter. Specifies the foreground color for the selection in the drop down filter window.
		Specifies the background color for the drop down

exBackColorFilter	26	filter window. Use the ColumnFilterButton property to specify whether the drop down filter bar button is visible or hidden.
exForeColorFilter	27	Specifies the foreground color for the drop down filter window. Use the ColumnFilterButton property to specify whether the drop down filter bar button is visible or hidden.
exCursorHoverColumn	32	Specifies the visual appearance for the column when the cursor hovers the column. By default, the exCursorHoverColumn property is zero, and it has no effect, so the visual appearance for the column is not changed when the cursor hovers the header.
exHeaderFilterBarActive	41	Specifies the visual appearance of the drop down filter bar button, while filter is applied to the column.
exCheckBoxState0	70	Specifies the visual appearance for the check box in 0 state (unchecked).
exCheckBoxState1	71	Specifies the visual appearance for the check box in 1 state (checked).
exCheckBoxState2	72	Specifies the visual appearance for the check box in 2 state (partial, not used).
exSelBackColorHide	166	Specifies the selection's background color, when the control has no focus.
exSelForeColorHide	167	Specifies the selection's foreground color, when the control has no focus.
exTreeGlyphOpen	180	Specifies the visual appearance for the +/- buttons when it is collapsed.
exTreeGlyphClose	181	Specifies the visual appearance for the +/- buttons when it is expanded.
exColumnsPositionSign	182	Specifies the visual appearance for the position sign between columns, when the user changes the position of the column by drag an drop.
exTreeLinesColor	186	Specifies the color to show the tree-lines (connecting lines from the parent to the children)
exVSUp	256	The up button in normal state.
exVSUpP	257	The up button when it is pressed.
exVSUpD	258	The up button when it is disabled.
exVSUpH	259	The up button when the cursor hovers it.

exVSThumb	260	The thumb part (exThumbPart) in normal state.
exVSThumbP	261	The thumb part (exThumbPart) when it is pressed.
exVSThumbD	262	The thumb part (exThumbPart) when it is disabled.
exVSThumbH	263	The thumb part (exThumbPart) when cursor hovers it.
exVSDown	264	The down button in normal state.
exVSDownP	265	The down button when it is pressed.
exVSDownD	266	The down button when it is disabled.
exVSDownH	267	The down button when the cursor hovers it.
exVSLower	268	The lower part (exLowerBackPart) in normal state.
exVSLowerP	269	The lower part (exLowerBackPart) when it is pressed.
exVSLowerD	270	The lower part (exLowerBackPart) when it is disabled.
exVSLowerH	271	The lower part (exLowerBackPart) when the cursor hovers it.
exVSUpper	272	The upper part (exUpperBackPart) in normal state.
exVSUpperP	273	The upper part (exUpperBackPart) when it is pressed.
exVSUpperD	274	The upper part (exUpperBackPart) when it is disabled.
exVSUpperH	275	The upper part (exUpperBackPart) when the cursor hovers it.
exVSBack	276	The background part (exLowerBackPart and exUpperBackPart) in normal state.
exVSBackP	277	The background part (exLowerBackPart and exUpperBackPart) when it is pressed.
exVSBackD	278	The background part (exLowerBackPart and exUpperBackPart) when it is disabled.
exVSBackH	279	The background part (exLowerBackPart and exUpperBackPart) when the cursor hovers it.
exHSLeft	384	The left button in normal state.

exHSLeftP	385	The left button when it is pressed.
exHSLeftD	386	The left button when it is disabled.
exHSLeftH	387	The left button when the cursor hovers it.
exHSThumb	388	The thumb part (exThumbPart) in normal state.
exHSThumbP	389	The thumb part (exThumbPart) when it is pressed.
exHSThumbD	390	The thumb part (exThumbPart) when it is disabled.
exHSThumbH	391	The thumb part (exThumbPart) when the cursor hovers it.
exHSRight	392	The right button in normal state.
exHSRightP	393	The right button when it is pressed.
exHSRightD	394	The right button when it is disabled.
exHSRightH	395	The right button when the cursor hovers it.
exHSLower	396	The lower part (exLowerBackPart) in normal state.
exHSLowerP	397	The lower part (exLowerBackPart) when it is pressed.
exHSLowerD	398	The lower part (exLowerBackPart) when it is disabled.
exHSLowerH	399	The lower part (exLowerBackPart) when the cursor hovers it.
exHSUpper	400	The upper part (exUpperBackPart) in normal state.
exHSUpperP	401	The upper part (exUpperBackPart) when it is pressed.
exHSUpperD	402	The upper part (exUpperBackPart) when it is disabled.
exHSUpperH	403	The upper part (exUpperBackPart) when the cursor hovers it.
exHSBack	404	The background part (exLowerBackPart and exUpperBackPart) in normal state.
exHSBackP	405	The background part (exLowerBackPart and exUpperBackPart) when it is pressed.
exHSBackD	406	The background part (exLowerBackPart and exUpperBackPart) when it is disabled.
exHSBackH	407	The background part (exLowerBackPart and exUpperBackPart) when the cursor hovers it.

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

constants ChangeEnum

Specifies the state for a file or a folder. Use State property to get the change state. The file or folder's state is valid while the file object is contained by collection passed by the Change event. Use the Folder property to specify whether the File object refers a file or a folder.

Name	Value	Description
Unchanged	0	The file or the folder is unchanged.
Changed	1	The file or the folder is changed.
Added	2	The file or the folder is added
Deleted	3	The file or the folder is deleted.

constants CheckBoxEnum

The CheckBoxEnum expression defines the type of check boxes that control supports. Use the HasCheckBox property to assign a check box for each item in your control. /*not supported in the lite version*/

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

constants DescriptionTypeEnum

The control's Description property defines descriptions for few control parts

Name	Value	Description
exFilterBarAll	0	Defines the caption of (All) in the filter bar window.
exFilterBarFilterForCaption	1	Defines the caption of "Filter For:" in the filter bar window.
exFilterBarFilterTitle	2	Defines the title for the filter tooltip.
exFilterBarPatternFilterTitle	3	Defines the title for the filter pattern tooltip.
exFilterBarTooltip	4	Defines the tooltip for drop down filter window.
exFilterBarPatternTooltip	5	Defines the tooltip for drop down filter pattern field.
exFilterBarFilterForTooltip	6	Defines the tooltip for "Filter For:" window.
exFilterBarExclude	7	Specifies the 'Exclude' caption being displayed in the drop down filter.

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 RFT 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 in OLE drag and drop functionality
exOLEDropManual	1	The ExFileView 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 the OLE drag and drop support: OLECompleteDrag, OLEDragDrop, OLEDragOver, OLEGiveFeedback, OLESetData, OLEStartDrag.

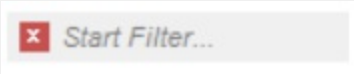
constants FileColumnEnum

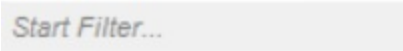
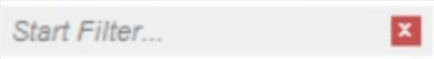
The FileColumnEnum type specifies the columns into the file-view control. You can use the ColumnsVisible property to show/hide multiple columns at once. The FileColumnEnum type supports the following flags:

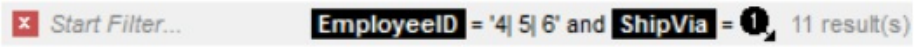
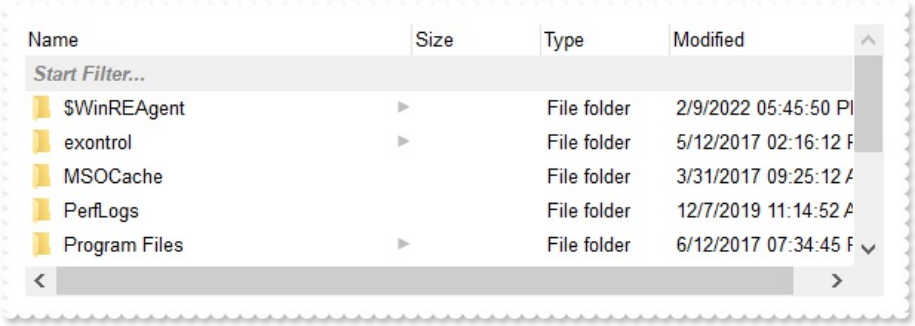
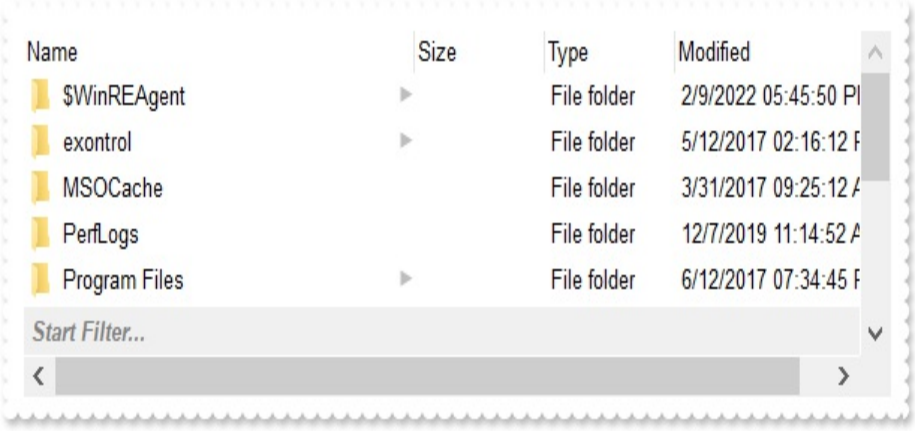
Name	Value	Description
exFileColumnName	2	Indicates the Name column.
exFileColumnSize	4	Indicates the Size column.
exFileColumnType	8	Indicates the Type column.
exFileColumnModified	16	Indicates the Modified column.

constants FilterBarVisibleEnum

The FilterBarVisibleEnum type defines the flags you can use on FilterBarPromptVisible property. The FilterBarCaption property defines the caption to be displayed on the control's filter bar. 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.
exFilterBarPromptVisible	1	The exFilterBarPromptVisible flag specifies that the control's filter bar displays the filter prompt. The exFilterBarPromptVisible, exFilterBarVisible, exFilterBarCaptionVisible flag , forces the control's filter-prompt, filter bar or filter bar description (even empty) to be shown. If missing, no filter prompt is displayed. The FilterBarPrompt property to specify the HTML caption being displayed in the filter bar when the filter pattern is missing. 
exFilterBarVisible	2	The exFilterBarVisible flag forces the control's filter bar to be shown, no matter if any filter is applied. If missing, no filter bar is displayed while the control has no filter applied.  or combined with exFilterBarPromptVisible 
exFilterBarCaptionVisible	4	The exFilterBarVisible flag forces the control's filter bar to display the FilterBarCaption property. 

exFilterBarSingleLine	16	The exFilterBarVisible flag specifies that the caption on the control's filter bar id displayed on a single line. The exFilterBarSingleLine flag , specifies that the filter bar's caption is shown on a single line, so HTML tag or \r\n are not handled. By default, the control's filter description applies word wrapping. Can be combined to exFilterBarCompact to display a single-line filter bar. If missing, the caption on the control's filter bar is displayed on multiple lines. You can change the height of the control's filter bar using the FilterBarHeight property.
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. 
		The exFilterBarCompact flag compacts the control's filter bar, so the filter-prompt will be displayed to

exFilterBarCompact	2048	the left, while the control's filter bar caption will be displayed to the right. This flag has effect only if combined with the exFilterBarPromptVisible. This flag can be combined with the exFilterBarSingleLine flag, so all filter bar will be displayed compact and on a single line. 
exFilterBarTop	8192	The exFilterBarTop flag displays the filter-bar on top (between control's header and files section as shown:  By default, the filter-bar is shown aligned to the bottom (between files and horizontal-scroll bar) as shown: 

constants FilterIncludeEnum

Use the FilterInclude property to specify the items being included, when the list is filtered

Name	Value	Description
exItemsWithoutChilds	0	Filters items without including their child items.
exItemsWithChilds	1	Filters items including their child items.
exRootsWithoutChilds	2	Filters only root items without including their child items.
exRootsWithChilds	3	Filters only root items including their child items.
exMatchingItemsOnly	4	exMatchingItemsOnly. Only items that match the filter are included, no parents, no children.
exMatchIncludeParent	240	exMatchIncludeParent. If the value does not match, check if any parent has a matching value.

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 FilterBarPromptColumns specifies the list of columns to be used when filtering. 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 FilterTypeEnum

Defines the type of filter applies to a column. The ColumnFilterType property defines the filter's type on specified column.

Name	Value	Description
exAll	0	No filter applied
exPattern	1	Only items that match the pattern are included. The Filter property defines the pattern. A pattern may contain the wild card characters '?' for any single character, '*' for zero or more occurrences of any character, '#' for any digit character. If any of the *, ?, # or | characters are preceded by a \ (escape character) it masks the character itself.
exFilter	240	Only the items that are in the ColumnFilter property are included.

constants IncludeParentEnum

The IncludeParentEnum type specifies whether the control includes the parent folder. The IncludeParent property retrieves or sets a value that indicates whether the control includes the parent folder. The IncludeParentEnum type supports the following values:

Name	Value	Description
exNoIncludeParent	0	No parent folder is included.
exIncludeParent	1	The parent folder is included as a normal item.
exIncludeLockedParent	2	The parent folder is included as a locked item. A locked item is not scrolable.

constants IncrementalSearchEnum

The IncrementalSearchEnum type specifies how the control searches for objects while user type characters inside. An incremental search begins searching as soon as you type the first character of the search string. Use the IncrementalSearch property to specify how the control finds objects based on the typed characters.

Name	Value	Description
exDefaultStartWith	-1	Specifies that the control looks for objects that starts with typed characters, without highlighting the found result.
exStartWith	0	Specifies that the control looks for objects that starts with typed characters, with highlighting the found result.
exContains	1	Specifies that the control looks for objects that contains typed characters, with highlighting the found result.

constants OptionEnum

Use the Option property to change the control's options that are listed bellow. Use the Refresh method to refresh the control's content.

Name	Value	Description
exModifiedToday	0	Retrieves or sets a value that indicates the caption being displayed on the 'Modified' column if the file was changed today. By default, the Option(exModifiedToday) property is "today". String expression.
exModifiedDaysAgo	1	Retrieves or sets a value that indicates the caption being displayed on the 'Modified' column when the file was changed n-th days ago. By default, the Option(exModifiedDaysAgo) property is "%i day(s) ago". The string may contain a single %i expression that indicates the number of days that should be displayed. For instance, in German language it would be better if we could display "vor 10 Tagen" instead "10 day(s) ago", and so the Option(exModifiedDaysAgo) property should be "vor %i Tagen". String expression.
		Retrieves or sets a value that indicates the format of the date being displayed on the 'Modified' column. By default, the Option (exModifiedDateFormat) property is "M/d/yyyy ", it means that the date is being displayed as "10/13/2004". The exModifiedDateFormat option may include the following predefined strings: • **d** (Day of month as digits with no leading zero for single-digit days.) • **dd** (Day of month as digits with leading zero for single-digit days.) • **ddd** (Day of week as a three-letter abbreviation.) • **dddd** (Day of week as its full name.) • **M** (Month as digits with no leading zero for

exModifiedDateFormat	2	single-digit months.) • **MM** (Month as digits with leading zero for single-digit months.) • **MMM** (Month as a three-letter abbreviation.) • **MMMM** (Month as its full name.) • **y** (Year as last two digits, but with no leading zero for years less than 10.) • **yy** (Year as last two digits, but with leading zero for years less than 10.) • **yyyy** (Year represented by full four digits.) For instance, use the format "ddd, MMM dd yy" to get the date displayed as "Wed, Aug 31 94". String expression.
exModifiedTimeFormat	3	Retrieves or sets a value that indicates the format of the time being displayed on the 'Modified' column. By default, the Option(exModifiedTimeFormat) property is "hh:mm:ss tt", it means that the time is displayed as "03:45:12 PM". The exModifiedTimeFormat may include the following predefined strings: • **h** (Hours with no leading zero for single-digit hours; 12-hour clock.) • **hh** (Hours with leading zero for single-digit hours; 12-hour clock.) • **H** (Hours with no leading zero for single-digit hours; 24-hour clock.) • **HH** (Hours with leading zero for single-digit hours; 24-hour clock.) • **m** (Minutes with no leading zero for single-digit minutes) • **mm** (Minutes with no leading zero for single-digit minutes.) • **s** (Seconds with no leading zero for single-digit seconds.) • **ss** (Seconds with leading zero for single-digit seconds.) • **t** (One character time-marker string, such as A or P.)

		• **tt** (Multicharacter time-marker string, such as AM or PM.) String expression.
exHideFileExtensionsForKnownFileTypes	4	Hides or shows the three-letter file-name extensions for certain files, reducing clutter in folder windows. By default, the Option(exHideFileExtensionsForKnownFileTypes) property is False. If the Option(exHideFileExtensionsForKnownFileTypes) property is False, the control shows the extensions for any file. If the Option(exHideFileExtensionsForKnownFileTypes) property is True, the control displays the file name according to the Windows Explorer option "**Hide File Extensions For Known File Types**". For instance, if the Windows Explorer option "Hide File Extensions For Known File Types" is checked, the control control hides file extensions for known file types, else the file extension is visible. Boolean expression
exSizeFormat	5	Specifies the format to display the Size column. You can use the Option(exSizeFormat) property to indicate whether the Size column should displays the size of files in GB (GigaBytes), MB (MegaBytes), KB (KiloBytes) or Bytes. The Option(exSizeFormat) property can be a BIT combination of the following flags: • 1 indicates the file size in bytes (**Bytes**) • 2 indicates the file size in kilo-bytes (1 **KB** = 1024 Bytes) • 4 indicates the file size in mega-bytes (1 **MB** = 1024 KB) • 8 indicates the file size in giga-bytes (1 **GB** = 1024 MB) For instance, if the Option(exSizeFormat) is 1 + 2 + 4 + 8 (=15), the size column may display Bytes for files with the size less than 1 KB, Kilo-Bytes for files

	with the size less than 1 MB, Mega-Bytes for files with the size less than 1 GB, and GB files. In other words a file of 100 Bytes, will display 100 Bytes, or a file of 29,038,225,408 Bytes, will display 27.04 GB. By default the Option(exSizeFormat) is 2 (KB, in other words the size column is displayed in KB). Long expression.

constants PictureDisplayEnum

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

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

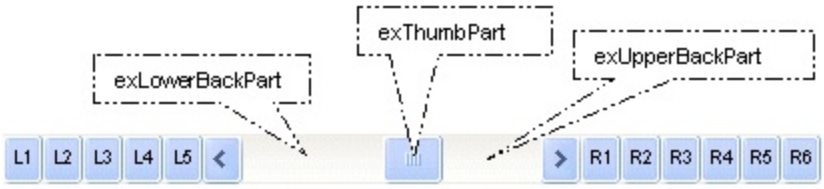
constants ScrollBarEnum

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

Name	Value	Description
exVScroll	0	Indicates the vertical scroll bar.
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. The control fires the ScrollButtonClick event when the user clicks any button in the control's scrollbar.



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

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

constants SearchStateEnum

The SearchStateEnum value specifies whether the searching files begins or ends. Use the Search property to search for files.

Name	Value	Description
StartSearching	0	The searching files starts.
EndSearching	1	The searching files ends.

constants StateChangeEnum

Specifies the new state of the control. Use the StateChange event to notify your application when the control's state is changed.

Name	Value	Description
RenameState	0	Fired when a file is renamed. This notification is not fired if an outside process change or renames a file in the current view. This notification is sent only if the file is renamed inside the current control.
SetFocusState	1	The control gains the focus.
KillFocusState	2	The control loses the focus.
SelChangeState	3	Fired when the control's selection is changed. Use the Get method to collect the selected files/folders.
BrowseChangeState	4	Occurs when the control browses a new folder. The BrowseFolderPath property indicates the folder being browsed.
RefreshState	5	Notifies the application once the Refresh method is invoked.
UpdateChangeState	6	Fired when the browsed folder suffers a change (like an outside process renamed, deleted or added a file).
BeforeFilterChangeState	7	Fired just before starting filtering the files. This notifies your application once the control starts filtering files and folders based on the UI actions.
AfterFilterChangeState	8	Fired after control has filtered the files. This notifies your application once the control starts filtering files and folders based on the UI actions.
BeforeLoadState	9	Notifies the application once the control starts loading the objects (the name of the objects).
AfterLoadState	10	Notifies the application once the control ends loading the objects (the name of the objects). Even if this state is fired, the control still can look for information about current files or folders until the ReadyState is sent.
BeforeExpandFolderState	11	Notifies the application once a folder is expanding. Use this notification to do your work before expanding an item.

AfterExpandFolderState	12	Notifies the application once a folder is expanded. Use this notification to do your work after an item is expanded.
BeforeCollapseFolderState	13	Notifies the application once a folder is collapsing. Use this notification to do your work before collapsing an item.
AfterCollapseFolderState	14	Notifies the application once a folder is collapsed. Use this notification to do your work after an item is collapsed.
CheckStateChange	15	Fired when the object's checkbox is changed.
BusyState	16	The BusyState event occurs once the control starts collecting information for current files and folders. This event notifies your application once the control start loading information about This state may occurs only if the Asynchronous property is set on true.
ReadyState	17	The ReadState event occurs once the control ends collecting information for current files and folders. This event notifies that the control is ready, in other words, all information about current view is loaded. This state may occurs only if the Asynchronous property is set on true.
StartFromToState	18	Occurs when the control starts applying the From/To format.
EndFromToState	19	Occurs when the control ends applying the From/To format.
ShowContextMenu	20	Occurs when the control is about to display the object'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 AllowMenuContext property specifies whether the control shows the object's context menu when the user presses the right click over a file or folder.
		Occurs when the control is about to execute a command from the object's context menu. The

ExecuteContextMenu	21	event occurs before executing the command selected from the context menu. 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 AllowMenuContext property specifies whether the control shows the object's context menu when the user presses the right click over a file or folder.
LoadingState	22	The LoadingState event occurs several time while the control loads files or folders.

The following VB sample enumerates the selected items:

```
Private Sub ExFileView1_StateChange(ByVal State As EXFILEVIEWLibCtl.StateChangeEnum)
   If State = SelChangeState Then
      Dim fs As Files, f As File
      Set fs = ExFileView1.Get(SelItems)
      For Each f In fs
         Debug.Print f.Name
      Next
   End If
End Sub
```

The following C++ sample enumerates the selected items:

```
void OnStateChangeExfileview1(long State)
{
   switch ( State )
   {
      case 0: /*StartSearching*/
      {
         OutputDebugString( "Start searching" );
         break;
      }
      case 1: /*EndSearching*/
      {
```

```
            OutputDebugString( "End searching" );
            break;
        }
    }
}
```

The following VB.NET sample enumerates the selected items:

```
Private Sub AxExFileView1_StateChange(ByVal sender As Object, ByVal e As
AxEXFILEVIEWLib._IExFileViewEvents_StateChangeEvent) Handles
AxExFileView1.StateChange
    Select Case e.state
        Case EXFILEVIEWLib.StateChangeEnum.SelChangeState
            With AxExFileView1.get_Get(EXFILEVIEWLib.TypeEnum.SelItems)
                Dim i As Integer
                For i = 0 To .Count - 1
                    With .Item(i)
                        Debug.WriteLine(.Name)
                    End With
                Next
            End With
    End Select
End Sub
```

The following C# sample enumerates the selected items:

```
private void axExFileView1_StateChange(object sender,
AxEXFILEVIEWLib._IExFileViewEvents_StateChangeEvent e)
{
    switch (e.state)
    {
        case EXFILEVIEWLib.StateChangeEnum.SelChangeState:
        {
            EXFILEVIEWLib.Files files =
axExFileView1.get_Get(EXFILEVIEWLib.TypeEnum.SelItems);
            for (int i = 0; i < files.Count; i++)
            {
                EXFILEVIEWLib.File file = files[i];
```

```
            System.Diagnostics.Debug.WriteLine(file.Name);
        }
        break;
    }
  }
}
```

The following VFP sample enumerates the selected items:

```
*** ActiveX Control Event ***
LPARAMETERS state

do case

 case state = 3 && SelChangeState
 with thisform.ExFileView1.Get( 0 ) && SelItems
   local i
   for i = 0 to .Count - 1
      with .Item(i)
         wait window nowait .Name
      endwith
   next
 endwith

endcase
```

constants TypeEnum

Specifies the type of objects that Get property retrieves. The item could be a file or a folder. Use the Folder property to specify whether an item (File object), is a file or a folder.

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

The following VB sample displays the list of files as they are displayed:

```
With ExFileView1.Get(VisibleItems)
   For i = 0 To .Count - 1
      With .Item(i)
         Debug.Print .Name
      End With
   Next
End With
```

The following C++ sample displays the list of files as they are displayed:

```
CFiles files = m_fileview.GetGet( 3 /*VisibleItems*/ );
for ( long i = 0; i < files.GetCount(); i++ )
   OutputDebugString( files.GetItem( COleVariant( i ) ).GetName() );
```

The following VB.NET sample displays the list of files as they are displayed:

```
With AxExFileView1.get_Get(EXFILEVIEWLib.TypeEnum.VisibleItems)
   Dim i As Integer
   For i = 0 To .Count - 1
      With .Item(i)
         Debug.WriteLine(.Name())
      End With
   Next
End With
```

The following C# sample displays the list of files as they are displayed:

```
EXFILEVIEWLib.Files files = axExFileView1.get_Get(EXFILEVIEWLib.TypeEnum.VisibleItems);
for (int i = 0; i < files.Count; i++)
{
   EXFILEVIEWLib.File file = files[i];
   System.Diagnostics.Debug.WriteLine(file.Name);
}
```

The following VFP sample displays the list of files as they are displayed:

```
With thisform.ExFileView1.Get(3)   && VisibleItems
   For i = 0 To .Count - 1
      With .Item(i)
         wait window nowait .Name
      EndWith
   Next
EndWith
```

constants UIVisualThemeEnum

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

Name	Value	Description
exNoVisualTheme	0	exNoVisualTheme
exDefaultVisualTheme	16777215	exDefaultVisualTheme
exHeaderVisualTheme	1	exHeaderVisualTheme
exFilterBarVisualTheme	2	exFilterBarVisualTheme
exButtonsVisualTheme	4	exButtonsVisualTheme
exCheckBoxVisualTheme	64	exCheckBoxVisualTheme

Appearance object

/*not supported in the lite version*/ The component lets the user changes its visual appearance using **skins**, each one providing an additional visual experience that enhances viewing pleasure. Skins are relatively easy to build and put on any part of the control. The Appearance object holds a collection of skins. The Appearance object supports the following properties and methods:

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

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

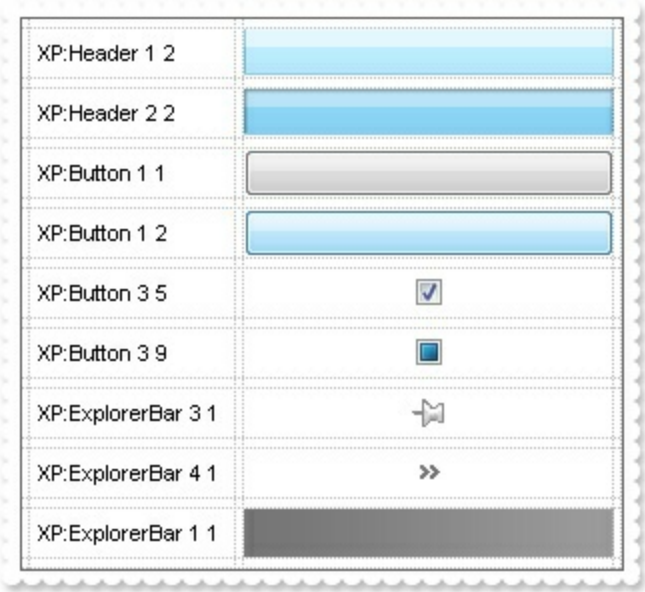
Adds or replaces a skin object to the control. /*not supported in the lite version*/

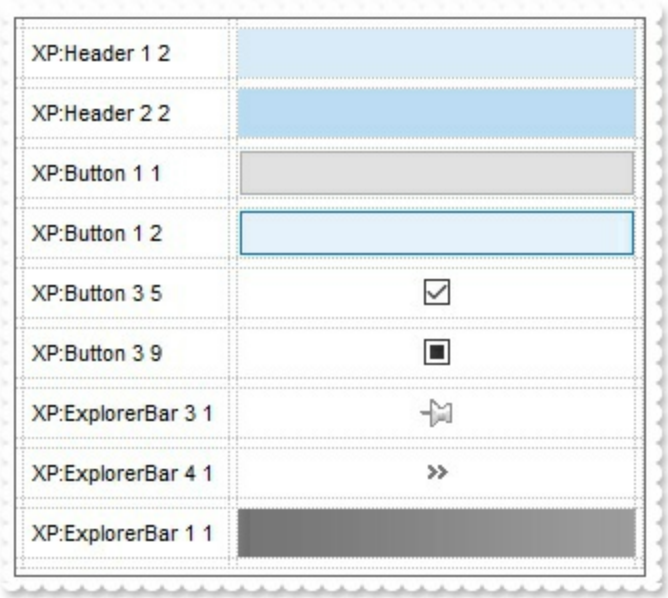
Type	Description
ID as Long	A Long expression that indicates the index of the skin being added or replaced. The value must be between 1 and 126, so Appearance collection should holds no more than 126 elements.
	The Skin parameter of the Add method can a STRING as explained bellow, a BYTE[] / safe arrays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

values for window/class, part and start are defined at the end of this document. For instance the "XP:Header 1 2" indicates the part 1 of the Header class in the state 2, in the current Windows XP theme.

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

Skin as Variant

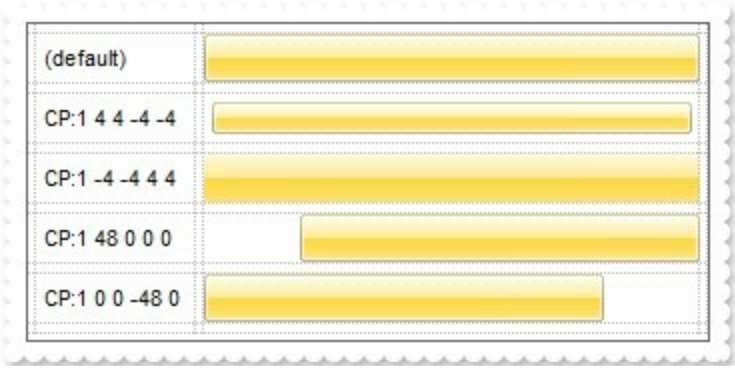




- A copy of another skin with different coordinates (position, size), if the Skin parameter starts with "**CP**:". Use this option, to display the EBN, using different coordinates (position, size). By default, the EBN skin object is rendered on the part's client area. Using this option, you can display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

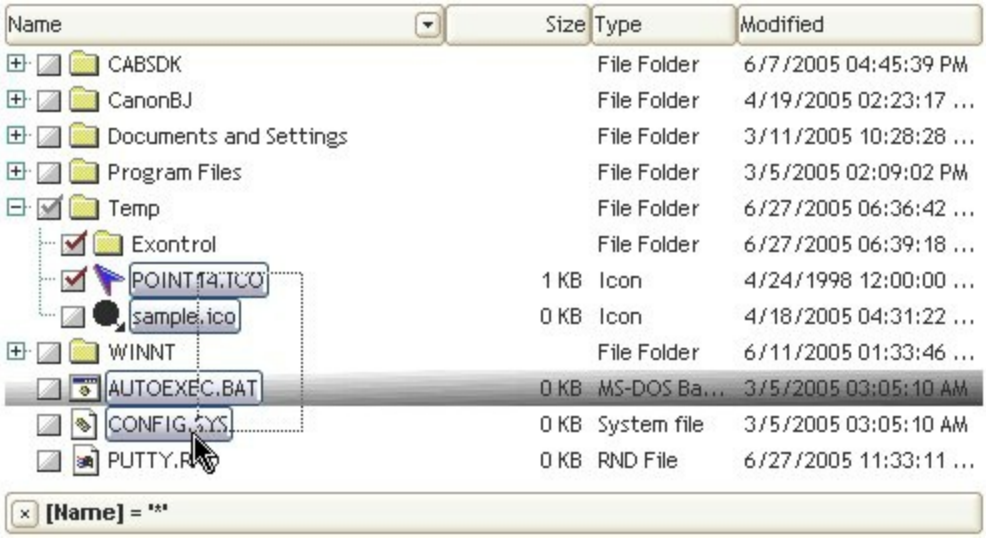
Skin parameter is: "CP:ID Left Top Right Bottom" where the ID is the identifier of the EBN to be used (it is a number that specifies the ID parameter of the Add method), Left, Top, Right and Bottom parameters/numbers specifies the relative position to the part's client area, where the EBN should be rendered.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

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



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

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



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

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

- selected file/folder, SelBackColor property
- control's **header bar**, BackColorHeader property
- control's **filter bar**, FilterBarBackColor property
- "**drop down**" filter bar button, "close" filter bar button, and so on, Background property

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

```
With ExFileView1
   With .VisualAppearance
      .Add &H23, App.Path + "\selected.ebn"
   End With
   .SelForeColor = RGB(0, 0, 0)
```

```
    .SelBackColor = &H23000000
End With
```

The sample adds the skin with the index 35 (Hexa 23), and applies to the selected item using the SelBackColor property.

The following C++ sample applies a new appearance to the selected item(s):

```
#include "Appearance.h"
m_fileview.GetVisualAppearance().Add( 0x23,
COleVariant(_T("D:\\Temp\\ExFileView_Help\\selected.ebn")) );
m_fileview.SetSelBackColor( 0x23000000 );
m_fileview.SetSelForeColor( 0 );
```

The following VB.NET sample applies a new appearance to the selected item(s):

```
With AxExFileView1
    With .VisualAppearance
        .Add(&H23, "D:\Temp\ExFileView_Help\selected.ebn")
    End With
    .SelForeColor = Color.Black
    .Template = "SelBackColor = 587202560"
End With
```

The VB.NET sample uses the Template property to assign a new value to the SelBackColor property. The 587202560 value represents &23000000 in hexadecimal.

The following C# sample applies a new appearance to the selected item(s):

```
axExFileView1.VisualAppearance.Add(0x23, "D:\\Temp\\ExFileView_Help\\selected.ebn");
axExFileView1.Template = "SelBackColor = 587202560";
```

The following VFP sample applies a new appearance to the selected item(s):

```
With thisform.ExFileView1
    With .VisualAppearance
        .Add(35, "D:\Temp\ExFileView_Help\selected.ebn")
    EndWith
    .SelForeColor = RGB(0, 0, 0)
    .SelBackColor = 587202560
```

```
EndWith
```

The 587202560 value represents &23000000 in hexadecimal. The 32 value represents &23 in hexadecimal

The screen shot was generated using the following template:

```
Appearance  = 0
ExpandFolders = True
IncludeFilesInFolder = True
HasCheckBox = 1

VisualAppearance
{
   ' Header
   Add(1,
"gBFLBCJwBAEHhEJAEGg4BcoDg6AABACAxWgKBADQKAAyDIKsEQGGIZRhhGIwAgaFIXQK

   ' HeaderFilterBarButton

Add(2,"gBFLBCJwBAEHhEJAEGg4BCwEg6AADACAxRDAMgBQKAAzQFAYZhxBaERiGIZ4JhUA



Add(3,"gBFLBCJwBAEHhEJAEGg4BFQEg6AADACAxRDAMgBQKAAzQFAYZhxBaERiGIZ4JhUA

   ' SelectedItem
   Add(4,
"gBFLBCJwBAEHhEJAEGg4BV4Fg6AABACAxWgKBADQKAAyDIKsEQGGIZRhhGIwAgaFIXQK



Add(6,"gBFLBCJwBAEHhEJAEGg4BKYEg6AADACAxRDAMgBQKAAzQFAYZhxBaERiGIZ4JhUA

}

'BackColor = 2147483652
BackColor = RGB(255,255,255)
BackColorHeader = 16777216          '0x01BBGGRR
FilterBarBackColor = 16777216       '0x01BBGGRR
```

```
Background(0) = 33554432            '0x02BBGGRR
Background(1) = 50331648            '0x03BBGGRR
SelBackColor = 67108864            '0x04BBGGRR
SelForeColor = 0
'ForeColorHeader = RGB(255,255,255)
ColumnFilterButton("Name") = True

FileTypes
{
   Add("*WIN*")
   {
      Folder = True
      BackColor = 100663296        '0x06BBGGRR
      Apply()
   }
}
```

method Appearance.Clear ()

Removes all skins in the control. /*not supported in the lite version*/

Type	Description

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

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

- selected file/folder, SelBackColor property
- control's **header bar**, BackColorHeader property
- control's **filter bar**, FilterBarBackColor property
- "**drop down**" filter bar button, "close" filter bar button, and so on, Background property

method Appearance.Remove (ID as Long)

Removes a specific skin from the control. /*not supported in the lite version*/

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

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

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

- selected file/folder, SelBackColor property
- control's **header bar**, BackColorHeader property
- control's **filter bar**, FilterBarBackColor property
- "**drop down**" filter bar button, "close" filter bar button, and so on, Background property

property Appearance.RenderType as Long

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

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

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

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

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

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

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

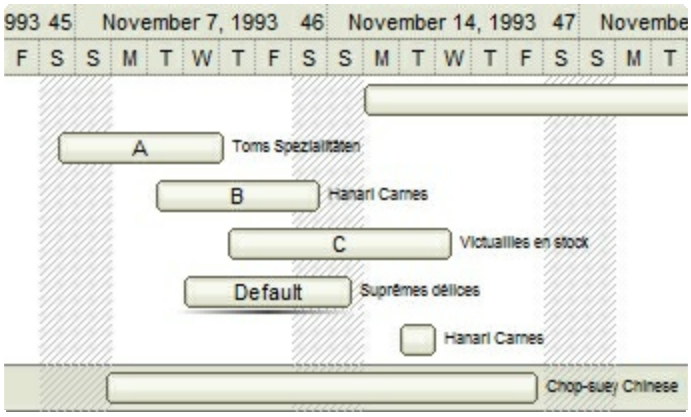
In the following screen shot the following objects displays the current EBN with a different color:

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

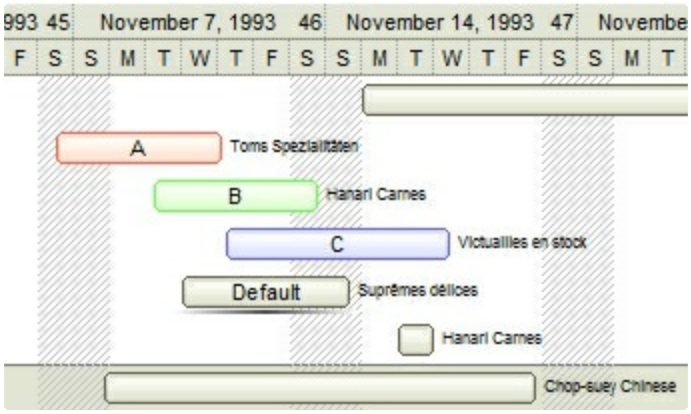
- "C" in Blue (RGB(0,0,255), for instance the bar's property exBarColor is 0x1FF0000
- "Default", no color is specified, for instance the bar's property exBarColor is 0x1000000

The RenderType property could be one of the following:

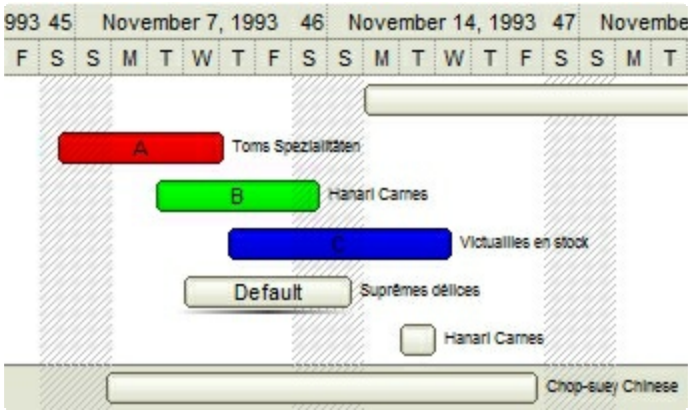
- ***-3**, no color is applied*. For instance, the BackColorHeader = &H1FF0000 is displayed as would be .BackColorHeader = &H1000000, so the 0xFF0000 color (Blue color) is ignored. You can use this option to allow the control displays the EBN colors or not.



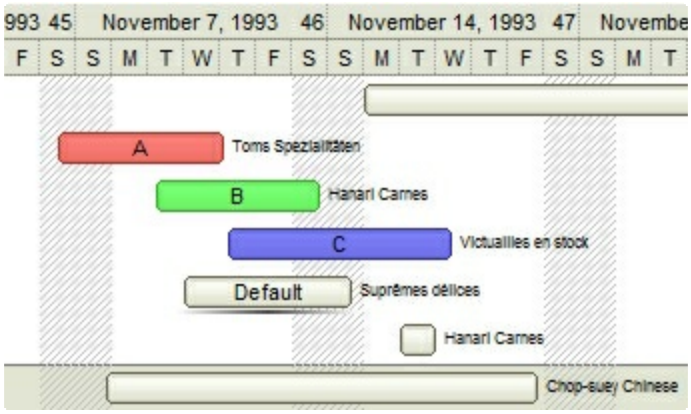
- ***-2**, OR-color scheme*. The color to be applied on the part of the control is a OR bit combination between the original EBN color and the specified color. For instance, the BackColorHeader = &H1FF0000, applies the OR bit for the entire Blue channel, or in other words, it applies a less Blue to the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



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

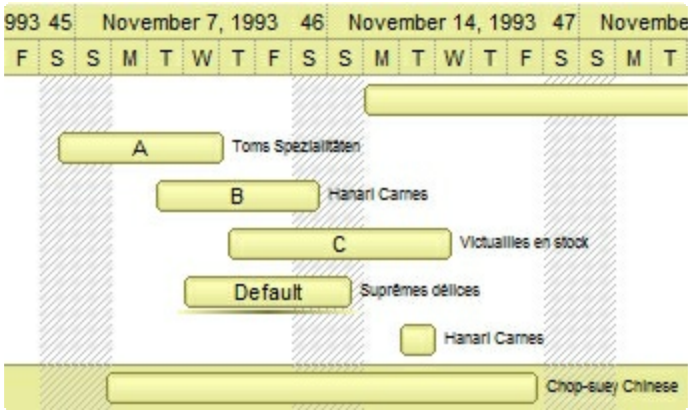


- ***0**, default*, the specified color is applied to the EBN. For instance, the BackColorHeader = &H1FF0000, applies a Blue color to the object. This option could be used to specify any color for the part of the components, that support EBN objects, not only solid colors.

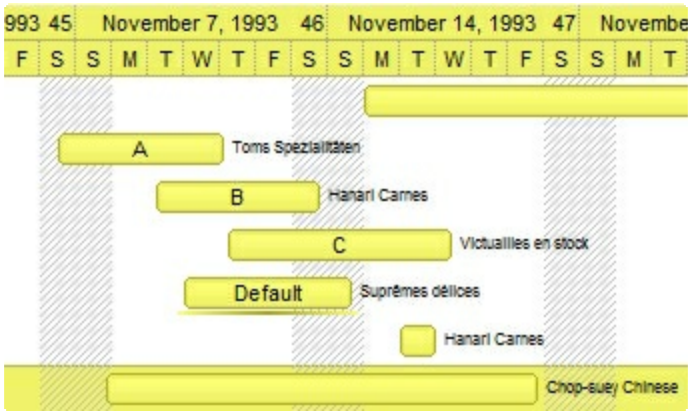


- **0xAABBGGRR**, where the AA a value between 0 to 255, which indicates the transparency, and RR, GG, BB the red, green and blue values. This option applies the same color to all parts that displays EBN objects, whit ignoring any specified color in the color property. For instance, the RenderType on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

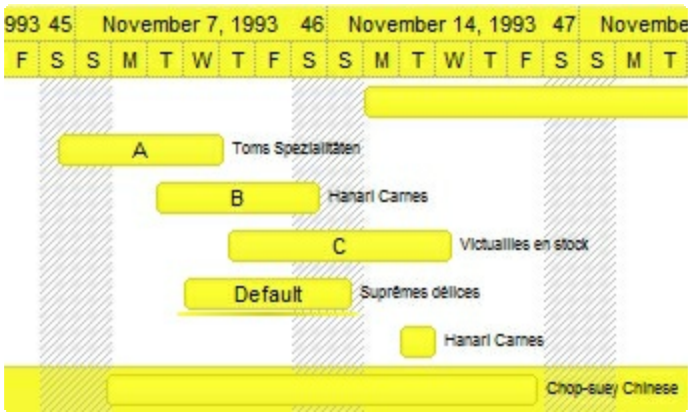
*The following picture shows the control with the RenderType property on 0x**40**00FFFF (25% Yellow, 0x40 or 64 in decimal is 25% from 256):*



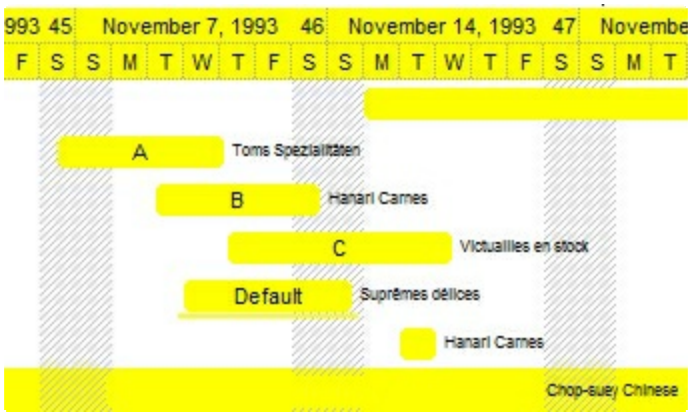
*The following picture shows the control with the RenderType property on 0x**80**00FFFF (50% Yellow, 0x80 or 128 in decimal is 50% from 256):*



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



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


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

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 ExDataObject object.

Type	Description

The Clear method can be called only for drag sources. The control fires the OLEStartDrag event to notify your application that the user stars dragging files. The OLEDragDrop event notifies your application that the user drags some data on the control.

The following VB sample starts dragging the selected files:

```
Private Sub ExFileView1_OLEStartDrag(ByVal Data As ExDataObject, AllowedEffects As
Long)
   Data.Files.Clear
   With ExFileView1.Get(SelItems)
      Dim i As Long
      For i = 0 To .Count - 1
       Data.Files.Add .Item(i).FullName
      Next
   End With
   If (Data.Files.Count > 0) Then
      AllowedEffects = 1
      Data.SetData , exCFFiles
   End If
End Sub
```

The following C++ sample starts dragging the selected files:

```
#import <exfilevw.dll>
void OnOLEStartDragExfileview1(LPDISPATCH Data, long FAR* AllowedEffects)
{
   EXFILEVIEWLib::IExDataObjectPtr spData( Data );
   spData->Clear();
   CFiles files = m_fileview.GetGet( 0 /*SelItems*/ );
   for ( long i = 0; i < files.GetCount(); i++ )
      spData->Files->Add( files.GetItem( COleVariant( i ) ).GetFullName().operator
LPCTSTR() );
   if ( spData->Files->Count > 0 )
   {
```

```
        *AllowedEffects = 1; /*exOLEDropEffectCopy*/
        spData->SetData( vtMissing, COleVariant( long(15) ) ); /*exCFFiles*/
    }
}
```

The C++ requires #import <exfilevw.dll> to import definitions for ExDataObject and ExDataObjectFiles objects. The #import <exfilevw.dll> generates the EXFILEVIEWLib namespace. If the exfilevw.dll file is located in other directory than system folder, the correct path should be provided, else a compiler error occurs.

The following VB.NET sample starts dragging the selected files:

```
Private Sub AxExFileView1_OLEStartDrag(ByVal sender As Object, ByVal e As
AxEXFILEVIEWLib._IExFileViewEvents_OLEStartDragEvent) Handles
AxExFileView1.OLEStartDrag
    e.data.Files.Clear()
    With AxExFileView1.get_Get(EXFILEVIEWLib.TypeEnum.SelItems)
        Dim i As Integer
        For i = 0 To .Count - 1
            e.data.Files.Add(.Item(i).FullName())
        Next
    End With
    If (e.data.Files.Count > 0) Then
        e.allowedEffects = 1
        e.data.SetData(, EXFILEVIEWLib.exClipboardFormatEnum.exCFFiles)
    End If
End Sub
```

The following C# sample starts dragging the selected files:

```
private void axExFileView1_OLEStartDrag(object sender,
AxEXFILEVIEWLib._IExFileViewEvents_OLEStartDragEvent e)
{
    e.data.Files.Clear();
    EXFILEVIEWLib.Files files = axExFileView1.get_Get(EXFILEVIEWLib.TypeEnum.SelItems);
    for ( int i = 0 ; i < files.Count; i++ )
        e.data.Files.Add(files[i].FullName);
    if (e.data.Files.Count > 0)
    {
```

```
        e.allowedEffects = 1;
        e.data.SetData(null, EXFILEVIEWLib.exClipboardFormatEnum.exCFFiles);
    }
}
```

The following VFP sample starts dragging the selected files:

```
*** ActiveX Control Event ***
LPARAMETERS data, allowedeffects

Data.Files.Clear
With thisform.ExFileView1.Get(0) && SelItems
   local i
   For i = 0 To .Count - 1
    data.Files.Add(.Item(i).FullName)
   Next
EndWith
If (Data.Files.Count > 0) Then
   AllowedEffects = 1
   data.SetData( , 15) && exCFFiles
EndIf
```

property ExDataObject.Files as ExDataObjectFiles

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

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

For instance, when the ExDataObject's format is exCFFiles, the Files property retrieves the files that were dropped to the ExFileView control. The control fires the OLEStartDrag event to notify your application that the user stars dragging files. The OLEDragDrop event notifies your application that the user drags some data on the control.

The following VB sample displays the list of files being dragged to the control (open your Windows Explorer, select some files and drag them to the control) :

```
Private Sub ExFileView1_OLEDragDrop(ByVal Data As EXFILEVIEWLibCtl.IExDataObject,
Effect As Long, ByVal Button As Integer, ByVal Shift As Integer, ByVal X As Single, ByVal Y As
Single)
   With Data.Files
      Dim i As Long
      For i = 0 To .Count - 1
         Debug.Print .Item(i)
      Next
   End With
End Sub
```

The following C++ sample displays the list of files being dragged to the control:

```
#import <exfilevw.dll>
void OnOLEDragDropExfileview1(LPDISPATCH Data, long FAR* Effect, short Button, short
Shift, long X, long Y)
{
   EXFILEVIEWLib::IExDataObjectPtr spData( Data );
   if ( spData )
   {
      EXFILEVIEWLib::IExDataObjectFilesPtr spFiles = spData->Files;
      for ( long i = 0; i < spFiles->Count; i++ )
         OutputDebugString( spFiles->Item[ i ] );
```

```
    }
}
```

The C++ requires #import <exfilevw.dll> to import definitions for ExDataObject and ExDataObjectFiles objects. The #import <exfilevw.dll> generates the EXFILEVIEWLib namespace. If the exfilevw.dll file is located in other directory than system folder, the correct path should be provided, else a compiler error occurs.

The following VB.NET sample displays the list of files being dragged to the control:

```
Private Sub AxExFileView1_OLEDragDrop(ByVal sender As Object, ByVal e As
AxEXFILEVIEWLib._IExFileViewEvents_OLEDragDropEvent) Handles
AxExFileView1.OLEDragDrop
    With e.data.Files
        Dim i As Long
        For i = 0 To .Count - 1
            Debug.WriteLine(.Item(i))
        Next
    End With
End Sub
```

The following C# sample displays the list of files being dragged to the control:

```
private void axExFileView1_OLEDragDrop(object sender,
AxEXFILEVIEWLib._IExFileViewEvents_OLEDragDropEvent e)
{
    EXFILEVIEWLib.ExDataObjectFiles files = e.data.Files;
    for (int i = 0; i < files.Count; i++)
        System.Diagnostics.Debug.WriteLine(files[i]);
}
```

The following VFP sample displays the list of files being dragged to the control:

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

With data.Files
    local i
    For i = 0 To .Count - 1
```

```
    wait window nowait .Item(i)
  Next
EndWith
```

method ExDataObject.GetData (Format as Integer)

Returns data from a ExDataObject 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 ExFileView 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 retrieve the filenames if the format of data is exCFiles. The OLEDragDrop event notifies your application that the user drags some data on the control. The control fires the OLEStartDrag event to notify your application that the user stars dragging files. The GetFormat property returns a value indicating whether the ExDataObject's data is of specified format

The following VB sample retrieves the text being dragged to the control:

```
Private Sub ExFileView1_OLEDragDrop(ByVal Data As EXFILEVIEWLibCtl.IExDataObject,
Effect As Long, ByVal Button As Integer, ByVal Shift As Integer, ByVal X As Single, ByVal Y As
Single)
   With Data
      If .GetFormat(EXFILEVIEWLibCtl.exClipboardFormatEnum.exCFText) Then
         MsgBox .GetData(EXFILEVIEWLibCtl.exClipboardFormatEnum.exCFText)
      End If
   End With
End Sub
```

The following C++ sample retrieves the text being dragged to the control:

```
#import <exfilevw.dll>
void OnOLEDragDropExfileview1(LPDISPATCH Data, long FAR* Effect, short Button, short
Shift, long X, long Y)
{
```

```
    EXFILEVIEWLib::IExDataObjectPtr spData( Data );
    if ( spData )
        if ( spData->GetFormat( 1 /*exCFText*/ ) )
        {
            CString strText = V2S( &spData->GetData( 1 /*exCFText*/ ) );
            MessageBox( strText );
        }
}
```

The C++ requires #import <exfilevw.dll> to import definitions for ExDataObject and ExDataObjectFiles objects. The #import <exfilevw.dll> generates the EXFILEVIEWLib namespace. If the exfilevw.dll file is located in other directory than system folder, the correct path should be provided, else a compiler error occurs.

The V2S function converts a VARIANT expression to a string, and may look like follows:

```
static CString V2S( VARIANT* pv, LPCTSTR szDefault = _T("") )
{
    if ( pv )
    {
        if ( pv->vt == VT_ERROR )
            return szDefault;

        COleVariant vt;
        vt.ChangeType( VT_BSTR, pv );
        return V_BSTR( &vt );
    }
    return szDefault;
}
```

The following VB.NET sample retrieves the text being dragged to the control:

```
Private Sub AxExFileView1_OLEDragDrop(ByVal sender As Object, ByVal e As
AxEXFILEVIEWLib._IExFileViewEvents_OLEDragDropEvent) Handles
AxExFileView1.OLEDragDrop
    With e.data
        If .GetFormat(EXFILEVIEWLib.exClipboardFormatEnum.exCFText) Then
            MessageBox.Show(.GetData(EXFILEVIEWLib.exClipboardFormatEnum.exCFText))
        End If
```

```
    End With
End Sub
```

The following C# sample retrieves the text being dragged to the control:

```
private void axExFileView1_OLEDragDrop(object sender,
AxEXFILEVIEWLib._IExFileViewEvents_OLEDragDropEvent e)
{
    if
(e.data.GetFormat(Convert.ToInt16(EXFILEVIEWLib.exClipboardFormatEnum.exCFText)))

MessageBox.Show(e.data.GetData(Convert.ToInt16(EXFILEVIEWLib.exClipboardFormatEnum

}
```

The following VFP sample retrieves the text being dragged to the control:

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

With data
    If .GetFormat( 1 ) Then && exCFText
        wait window nowait .GetData( 1 ) && exCFText
    EndIf
EndWith
```

method ExDataObject.GetFormat (Format as Integer)

Returns a value indicating whether the ExDataObject's data is of 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. Use the Files property to retrieves the filenames if the format of data is exCFiles. The OLEDragDrop event notifies your application that the user drags some data on the control. The control fires the OLEStartDrag event to notify your application that the user stars dragging files.

The following VB sample retrieves the text being dragged to the control:

```
Private Sub ExFileView1_OLEDragDrop(ByVal Data As EXFILEVIEWLibCtl.IExDataObject,
Effect As Long, ByVal Button As Integer, ByVal Shift As Integer, ByVal X As Single, ByVal Y As
Single)
   With Data
      If .GetFormat(EXFILEVIEWLibCtl.exClipboardFormatEnum.exCFText) Then
         MsgBox .GetData(EXFILEVIEWLibCtl.exClipboardFormatEnum.exCFText)
      End If
   End With
End Sub
```

The following C++ sample retrieves the text being dragged to the control:

```
#import <exfilevw.dll>
void OnOLEDragDropExfileview1(LPDISPATCH Data, long FAR* Effect, short Button, short
Shift, long X, long Y)
{
   EXFILEVIEWLib::IExDataObjectPtr spData( Data );
   if ( spData )
      if ( spData->GetFormat( 1 /*exCFText*/ ) )
      {
```

```
        CString strText = V2S( &spData->GetData( 1 /*exCFText*/ ) );
        MessageBox( strText );
    }
}
```

The C++ requires #import <exfilevw.dll> to import definitions for ExDataObject and ExDataObjectFiles objects. The #import <exfilevw.dll> generates the EXFILEVIEWLib namespace. If the exfilevw.dll file is located in other directory than system folder, the correct path should be provided, else a compiler error occurs.

The V2S function converts a VARIANT expression to a string, and may look like follows:

```
static CString V2S( VARIANT* pv, LPCTSTR szDefault = _T("") )
{
   if ( pv )
   {
      if ( pv->vt == VT_ERROR )
         return szDefault;

      COleVariant vt;
      vt.ChangeType( VT_BSTR, pv );
      return V_BSTR( &vt );
   }
   return szDefault;
}
```

The following VB.NET sample retrieves the text being dragged to the control:

```
Private Sub AxExFileView1_OLEDragDrop(ByVal sender As Object, ByVal e As
AxEXFILEVIEWLib._IExFileViewEvents_OLEDragDropEvent) Handles
AxExFileView1.OLEDragDrop
   With e.data
      If .GetFormat(EXFILEVIEWLib.exClipboardFormatEnum.exCFText) Then
         MessageBox.Show(.GetData(EXFILEVIEWLib.exClipboardFormatEnum.exCFText))
      End If
   End With
End Sub
```

The following C# sample retrieves the text being dragged to the control:

```
private void axExFileView1_OLEDragDrop(object sender,
AxEXFILEVIEWLib._IExFileViewEvents_OLEDragDropEvent e)
{
   if
(e.data.GetFormat(Convert.ToInt16(EXFILEVIEWLib.exClipboardFormatEnum.exCFText)))

MessageBox.Show(e.data.GetData(Convert.ToInt16(EXFILEVIEWLib.exClipboardFormatEnum

}
```

The following VFP sample retrieves the text being dragged to the control:

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

With data
   If .GetFormat( 1 ) Then && exCFText
      wait window nowait .GetData( 1 ) && exCFText
   EndIf
EndWith
```

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 being inserted to the 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. You can use the RegisterClipboardFormat API function to register a new clipboard format. This format can then be used as a valid clipboard format. The control fires the OLEStartDrag event to notify your application that the user stars dragging files. The OLEDragDrop event notifies your application that the user drags some data on the control. Use the Get property to retrieve the selected items. Use the FullName property to retrieve the full name of the file. Use the SingleSel property to allow multiple selection in the control.

The following VB sample starts dragging the selected files:

```
Private Sub ExFileView1_OLEStartDrag(ByVal Data As ExDataObject, AllowedEffects As
Long)
   Data.Files.Clear
   With ExFileView1.Get(SelItems)
      Dim i As Long
      For i = 0 To .Count - 1
       Data.Files.Add .Item(i).FullName
      Next
   End With
   If (Data.Files.Count > 0) Then
      AllowedEffects = 1
      Data.SetData , exCFFiles
   End If
End Sub
```

The following C++ sample starts dragging the selected files:

```
#import <exfilevw.dll>
void OnOLEStartDragExfileview1(LPDISPATCH Data, long FAR* AllowedEffects)
{
```

```
    EXFILEVIEWLib::IExDataObjectPtr spData( Data );
    spData->Clear();
    CFiles files = m_fileview.GetGet( 0 /*SelItems*/ );
    for ( long i = 0; i < files.GetCount(); i++ )
        spData->Files->Add( files.GetItem( COleVariant( i ) ).GetFullName().operator
LPCTSTR() );
    if ( spData->Files->Count > 0 )
    {
        *AllowedEffects = 1; /*exOLEDropEffectCopy*/
        spData->SetData( vtMissing, COleVariant( long(15) ) ); /*exCFFiles*/
    }
}
```

The C++ requires #import <exfilevw.dll> to import definitions for ExDataObject and ExDataObjectFiles objects. The #import <exfilevw.dll> generates the EXFILEVIEWLib namespace. If the exfilevw.dll file is located in other directory than system folder, the correct path should be provided, else a compiler error occurs.

The following VB.NET sample starts dragging the selected files:

```
Private Sub AxExFileView1_OLEStartDrag(ByVal sender As Object, ByVal e As
AxEXFILEVIEWLib._IExFileViewEvents_OLEStartDragEvent) Handles
AxExFileView1.OLEStartDrag
    e.data.Files.Clear()
    With AxExFileView1.get_Get(EXFILEVIEWLib.TypeEnum.SelItems)
        Dim i As Integer
        For i = 0 To .Count - 1
            e.data.Files.Add(.Item(i).FullName())
        Next
    End With
    If (e.data.Files.Count > 0) Then
        e.allowedEffects = 1
        e.data.SetData(, EXFILEVIEWLib.exClipboardFormatEnum.exCFFiles)
    End If
End Sub
```

The following C# sample starts dragging the selected files:

```
private void axExFileView1_OLEStartDrag(object sender,
```

```
AxEXFILEVIEWLib._IExFileViewEvents_OLEStartDragEvent e)
{
   e.data.Files.Clear();
   EXFILEVIEWLib.Files files = axExFileView1.get_Get(EXFILEVIEWLib.TypeEnum.SelItems);
   for ( int i = 0 ; i < files.Count; i++ )
      e.data.Files.Add(files[i].FullName);
   if (e.data.Files.Count > 0)
   {
      e.allowedEffects = 1;
      e.data.SetData(null, EXFILEVIEWLib.exClipboardFormatEnum.exCFFiles);
   }
}
```

The following VFP sample starts dragging the selected files:

```
*** ActiveX Control Event ***
LPARAMETERS data, allowedeffects

Data.Files.Clear
With thisform.ExFileView1.Get(0) && SelItems
   local i
   For i = 0 To .Count - 1
    data.Files.Add(.Item(i).FullName)
   Next
EndWith
If (Data.Files.Count > 0) Then
   AllowedEffects = 1
   data.SetData( , 15) && exCFFiles
EndIf
```

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 the drag and drop data source. Use the Files property to retrieve the filenames if the format of data is exCFiles. The OLEDragDrop event notifies your application that the user drags some data on the control. The control fires the OLEStartDrag event to notify your application that the user stars dragging files. Use the Get property to retrieve the selected items. Use the FullName property to retrieve the full name of the file. You can use the RegisterClipboardFormat API function to register a new clipboard format. This format can then be used as a valid clipboard format. Use the SingleSel property to allow multiple selection in the control.

The following VB sample starts dragging the selected files:

```
Private Sub ExFileView1_OLEStartDrag(ByVal Data As ExDataObject, AllowedEffects As
Long)
   Data.Files.Clear
   With ExFileView1.Get(SelItems)
      Dim i As Long
      For i = 0 To .Count - 1
       Data.Files.Add .Item(i).FullName
      Next
   End With
   If (Data.Files.Count > 0) Then
      AllowedEffects = 1
      Data.SetData , exCFFiles
   End If
End Sub
```

The following C++ sample starts dragging the selected files:

```
#import <exfilevw.dll>
void OnOLEStartDragExfileview1(LPDISPATCH Data, long FAR* AllowedEffects)
{
   EXFILEVIEWLib::IExDataObjectPtr spData( Data );
   spData->Clear();
```

```
    CFiles files = m_fileview.GetGet( 0 /*SelItems*/ );
    for ( long i = 0; i < files.GetCount(); i++ )
        spData->Files->Add( files.GetItem( COleVariant( i ) ).GetFullName().operator
LPCTSTR() );
    if ( spData->Files->Count > 0 )
    {
        *AllowedEffects = 1; /*exOLEDropEffectCopy*/
        spData->SetData( vtMissing, COleVariant( long(15) ) ); /*exCFFiles*/
    }
}
```

The C++ requires #import <exfilevw.dll> to import definitions for ExDataObject and ExDataObjectFiles objects. The #import <exfilevw.dll> generates the EXFILEVIEWLib namespace. If the exfilevw.dll file is located in other directory than system folder, the correct path should be provided, else a compiler error occurs.

The following VB.NET sample starts dragging the selected files:

```
Private Sub AxExFileView1_OLEStartDrag(ByVal sender As Object, ByVal e As
AxEXFILEVIEWLib._IExFileViewEvents_OLEStartDragEvent) Handles
AxExFileView1.OLEStartDrag
    e.data.Files.Clear()
    With AxExFileView1.get_Get(EXFILEVIEWLib.TypeEnum.SelItems)
        Dim i As Integer
        For i = 0 To .Count - 1
            e.data.Files.Add(.Item(i).FullName())
        Next
    End With
    If (e.data.Files.Count > 0) Then
        e.allowedEffects = 1
        e.data.SetData(, EXFILEVIEWLib.exClipboardFormatEnum.exCFFiles)
    End If
End Sub
```

The following C# sample starts dragging the selected files:

```
private void axExFileView1_OLEStartDrag(object sender,
AxEXFILEVIEWLib._IExFileViewEvents_OLEStartDragEvent e)
{
```

```
    e.data.Files.Clear();
    EXFILEVIEWLib.Files files = axExFileView1.get_Get(EXFILEVIEWLib.TypeEnum.SelItems);
    for ( int i = 0 ; i < files.Count; i++ )
        e.data.Files.Add(files[i].FullName);
    if (e.data.Files.Count > 0)
    {
        e.allowedEffects = 1;
        e.data.SetData(null, EXFILEVIEWLib.exClipboardFormatEnum.exCFFiles);
    }
}
```

The following VFP sample starts dragging the selected files:

```
*** ActiveX Control Event ***
LPARAMETERS data, allowedeffects

Data.Files.Clear
With thisform.ExFileView1.Get(0) && SelItems
    local i
    For i = 0 To .Count - 1
     data.Files.Add(.Item(i).FullName)
    Next
EndWith
If (Data.Files.Count > 0) Then
    AllowedEffects = 1
    data.SetData( , 15) && exCFFiles
EndIf
```

method ExDataObjectFiles.Clear ()

Removes all file names in the collection.

Type	Description

Use the Clear method to remove all filenames from the collection. Use the Add method to add new files to the drag and drop data source. Use the Files property to retrieves the filenames if the format of data is exCFiles. The OLEDragDrop event notifies your application that the user drags some data on the control. The control fires the OLEStartDrag event to notify your application that the user stars dragging files. Use the Get property to retrieve the selected items. Use the FullName property to retrieve the full name of the file. You can use the RegisterClipboardFormat API function to register a new clipboard format. This format can then be used as a valid clipboard format. Use the SingleSel property to allow multiple selection in the control.

The following VB sample starts dragging the selected files:

```
Private Sub ExFileView1_OLEStartDrag(ByVal Data As ExDataObject, AllowedEffects As
Long)
   Data.Files.Clear
   With ExFileView1.Get(SelItems)
      Dim i As Long
      For i = 0 To .Count - 1
       Data.Files.Add .Item(i).FullName
      Next
   End With
   If (Data.Files.Count > 0) Then
      AllowedEffects = 1
      Data.SetData , exCFFiles
   End If
End Sub
```

The following C++ sample starts dragging the selected files:

```
#import <exfilevw.dll>
void OnOLEStartDragExfileview1(LPDISPATCH Data, long FAR* AllowedEffects)
{
   EXFILEVIEWLib::IExDataObjectPtr spData( Data );
   spData->Clear();
```

```
    CFiles files = m_fileview.GetGet( 0 /*SelItems*/ );
    for ( long i = 0; i < files.GetCount(); i++ )
        spData->Files->Add( files.GetItem( COleVariant( i ) ).GetFullName().operator
LPCTSTR() );
    if ( spData->Files->Count > 0 )
    {
        *AllowedEffects = 1; /*exOLEDropEffectCopy*/
        spData->SetData( vtMissing, COleVariant( long(15) ) ); /*exCFFiles*/
    }
}
```

The C++ requires #import <exfilevw.dll> to import definitions for ExDataObject and ExDataObjectFiles objects. The #import <exfilevw.dll> generates the EXFILEVIEWLib namespace. If the exfilevw.dll file is located in other directory than system folder, the correct path should be provided, else a compiler error occurs.

The following VB.NET sample starts dragging the selected files:

```
Private Sub AxExFileView1_OLEStartDrag(ByVal sender As Object, ByVal e As
AxEXFILEVIEWLib._IExFileViewEvents_OLEStartDragEvent) Handles
AxExFileView1.OLEStartDrag
    e.data.Files.Clear()
    With AxExFileView1.get_Get(EXFILEVIEWLib.TypeEnum.SelItems)
        Dim i As Integer
        For i = 0 To .Count - 1
            e.data.Files.Add(.Item(i).FullName())
        Next
    End With
    If (e.data.Files.Count > 0) Then
        e.allowedEffects = 1
        e.data.SetData(, EXFILEVIEWLib.exClipboardFormatEnum.exCFFiles)
    End If
End Sub
```

The following C# sample starts dragging the selected files:

```
private void axExFileView1_OLEStartDrag(object sender,
AxEXFILEVIEWLib._IExFileViewEvents_OLEStartDragEvent e)
{
```

```
    e.data.Files.Clear();
    EXFILEVIEWLib.Files files = axExFileView1.get_Get(EXFILEVIEWLib.TypeEnum.SelItems);
    for ( int i = 0 ; i < files.Count; i++ )
        e.data.Files.Add(files[i].FullName);
    if (e.data.Files.Count > 0)
    {
        e.allowedEffects = 1;
        e.data.SetData(null, EXFILEVIEWLib.exClipboardFormatEnum.exCFFiles);
    }
}
```

The following VFP sample starts dragging the selected files:

```
*** ActiveX Control Event ***
LPARAMETERS data, allowedeffects

Data.Files.Clear
With thisform.ExFileView1.Get(0) && SelItems
    local i
    For i = 0 To .Count - 1
     data.Files.Add(.Item(i).FullName)
    Next
EndWith
If (Data.Files.Count > 0) Then
    AllowedEffects = 1
    data.SetData( , 15) && exCFFiles
EndIf
```

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.

Use the Count property to retrieve the number of files in the drag and drop data source. Use the Item property to retrieve the file giving its index. Use the Files property to retrieve the filenames if the format of data is exCFiles. The OLEDragDrop event notifies your application that the user drags some data on the control. Use the GetFormat property to retrieve the type of data being carried by the drag and drop data source.

The following VB sample displays the list of files being dragged to the control (open your Windows Explorer, select some files and drag them to the control) :

```
Private Sub ExFileView1_OLEDragDrop(ByVal Data As EXFILEVIEWLibCtl.IExDataObject,
Effect As Long, ByVal Button As Integer, ByVal Shift As Integer, ByVal X As Single, ByVal Y As
Single)
   With Data.Files
      Dim i As Long
      For i = 0 To .Count - 1
         Debug.Print .Item(i)
      Next
   End With
End Sub
```

The following C++ sample displays the list of files being dragged to the control:

```
#import <exfilevw.dll>
void OnOLEDragDropExfileview1(LPDISPATCH Data, long FAR* Effect, short Button, short
Shift, long X, long Y)
{
   EXFILEVIEWLib::IExDataObjectPtr spData( Data );
   if ( spData )
   {
      EXFILEVIEWLib::IExDataObjectFilesPtr spFiles = spData->Files;
      for ( long i = 0; i < spFiles->Count; i++ )
         OutputDebugString( spFiles->Item[ i ] );
```

```
    }
}
```

The C++ requires #import <exfilevw.dll> to import definitions for ExDataObject and ExDataObjectFiles objects. The #import <exfilevw.dll> generates the EXFILEVIEWLib namespace. If the exfilevw.dll file is located in other directory than system folder, the correct path should be provided, else a compiler error occurs.

The following VB.NET sample displays the list of files being dragged to the control:

```
Private Sub AxExFileView1_OLEDragDrop(ByVal sender As Object, ByVal e As
AxEXFILEVIEWLib._IExFileViewEvents_OLEDragDropEvent) Handles
AxExFileView1.OLEDragDrop
    With e.data.Files
        Dim i As Long
        For i = 0 To .Count - 1
            Debug.WriteLine(.Item(i))
        Next
    End With
End Sub
```

The following C# sample displays the list of files being dragged to the control:

```
private void axExFileView1_OLEDragDrop(object sender,
AxEXFILEVIEWLib._IExFileViewEvents_OLEDragDropEvent e)
{
    EXFILEVIEWLib.ExDataObjectFiles files = e.data.Files;
    for (int i = 0; i < files.Count; i++)
        System.Diagnostics.Debug.WriteLine(files[i]);
}
```

The following VFP sample displays the list of files being dragged to the control:

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

With data.Files
    local i
    For i = 0 To .Count - 1
```

```
        wait window nowait .Item(i)
    Next
EndWith
```

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

The Item property gets a file giving its index. The Count property counts the number of files in the collection. Use the Files property to retrieve the filenames if the format of data is exCFiles. The OLEDragDrop event notifies your application that the user drags some data on the control. Use the GetFormat property to retrieve the type of data being carried by the drag and drop data source.

The following VB sample displays the list of files being dragged to the control (open your Windows Explorer, select some files and drag them to the control) :

```
Private Sub ExFileView1_OLEDragDrop(ByVal Data As EXFILEVIEWLibCtl.IExDataObject,
Effect As Long, ByVal Button As Integer, ByVal Shift As Integer, ByVal X As Single, ByVal Y As
Single)
   With Data.Files
      Dim i As Long
      For i = 0 To .Count - 1
         Debug.Print .Item(i)
      Next
   End With
End Sub
```

The following C++ sample displays the list of files being dragged to the control:

```
#import <exfilevw.dll>
void OnOLEDragDropExfileview1(LPDISPATCH Data, long FAR* Effect, short Button, short
Shift, long X, long Y)
{
   EXFILEVIEWLib::IExDataObjectPtr spData( Data );
   if ( spData )
   {
      EXFILEVIEWLib::IExDataObjectFilesPtr spFiles = spData->Files;
      for ( long i = 0; i < spFiles->Count; i++ )
```

```
        OutputDebugString( spFiles->Item[ i ] );
    }
}
```

The C++ requires #import <exfilevw.dll> to import definitions for ExDataObject and ExDataObjectFiles objects. The #import <exfilevw.dll> generates the EXFILEVIEWLib namespace. If the exfilevw.dll file is located in other directory than system folder, the correct path should be provided, else a compiler error occurs.

The following VB.NET sample displays the list of files being dragged to the control:

```
Private Sub AxExFileView1_OLEDragDrop(ByVal sender As Object, ByVal e As
AxEXFILEVIEWLib._IExFileViewEvents_OLEDragDropEvent) Handles
AxExFileView1.OLEDragDrop
    With e.data.Files
        Dim i As Long
        For i = 0 To .Count - 1
            Debug.WriteLine(.Item(i))
        Next
    End With
End Sub
```

The following C# sample displays the list of files being dragged to the control:

```
private void axExFileView1_OLEDragDrop(object sender,
AxEXFILEVIEWLib._IExFileViewEvents_OLEDragDropEvent e)
{
    EXFILEVIEWLib.ExDataObjectFiles files = e.data.Files;
    for (int i = 0; i < files.Count; i++)
        System.Diagnostics.Debug.WriteLine(files[i]);
}
```

The following VFP sample displays the list of files being dragged to the control:

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

With data.Files
    local i
```

```
    For i = 0 To .Count - 1
        wait window nowait .Item(i)
    Next
EndWith
```

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 the Remove method to remove a file name from the collection. Use Clear method to remove all filenames. Use Add method to add your files to the drag and drop data source. The control fires the OLEStartDrag event to notify your application that the user stars dragging files. Use the Get property to retrieve the selected items. Use the FullName property to retrieve the full name of the file. You can use the RegisterClipboardFormat API function to register a new clipboard format. This format can then be used as a valid clipboard format. Use the SingleSel property to allow multiple selection in the control.

ExFileView 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: {F26C97E5-3E86-4CE4-935B-A997AB3DDBE4}. The object's program identifier is: "Exontrol.ExFileView". The /COM object module is: "ExFileVw.dll"

Provide rich display of file and folder information from within your applications. ExFileView is an ActiveX component for creating Windows Explorer-style functionality. Files with different attributes can be displayed with different color, background color, font, etc. It can also filter the files based on files extensions using Include or Exclude clauses. The ExFileView component is able to change the displayed icon, or type file, supports Drag & Drop, incremental search, mouse wheel and more. The ExFileView control is able to show the folders that contains files change into a given interval. The ExFileView components supports the following methods and properties:

Name	Description
AddColumnCustomFilter	Adds a custom filter to the column.
AllowEnterFolder	Specifies whether a new folder is opened, once the user presses the Enter key or double-clicks a folder.
AllowMenuContext	Enables or disables the file's context menu.
AllowRename	Retrieves or sets a value that indicates whether the control allows renaming items.
AllowSelectNothing	Specifies whether the current selection is erased, once the user clicks outside of the items section.
AllowShortcutFolders	Specifies whether the shortcut-folders are shown as folders or files.
Appearance	Retrieves or sets the control's appearance.
ApplyFilter	Applies the filter.
Asynchronous	Specifies whether the files and folders information is loading in the background.
AttachTemplate	Attaches a script to the current object, including the events, from a string, file, a safe array of bytes.
AutoDrag	Gets or sets a value that indicates the way the component supports the AutoDrag feature.
AutoUpdate	Determines whether the control is refreshed while a file or folder is changed, moved, or renamed.
BackColor	Retrieves or sets the control's background.
BackColorHeader	Specifies the header's background color.
	Returns or sets a value that indicates the background

Background	color for parts in the control.
BeginUpdate	Prevents the control from painting until the EndUpdate method is called.
BrowseFolderPath	Retrieves or sets the browsed folder path.
ChangeNotification	Enables or disables control's notifications by firing Change event, whether the control's list is altered.
ClearColumnCustomFilters	Clears the list of column's custom filters.
ClearFilter	Clears the filter.
ClearImages	Clears the loaded images.
ColumnAutoResize	Returns or sets a value indicating whether the control will automatically size its visible columns to fit on the control's client width.
ColumnCaption	Specifies the column's caption.
ColumnFilter	Specifies the column's filter when filter type is exFilter.
ColumnFilterButton	Specifies a value that indicates whether the column displays the filter button.
ColumnFilterType	Specifies the column's filter type.
ColumnsAllowSizing	Retrieves or sets a value that indicates whether a user can resize columns at run-time.
ColumnsVisible	Indicates the columns being visible.
ColumnVisible	Retrieves or sets a value that indicates whether the column is visible or hidden.
ColumnWidth	Retrieves or sets a value that indicates the column's width.
Copy	Copies the control's content to the clipboard, in the EMF format.
CopyTo	Exports the control's view to an EMF file.
Debug	Displays information in debug mode.
DefaultItemHeight	Retrieves or sets a value that indicates the default item height.
Description	Changes descriptions for control objects.
DisplayFoldersInfo	Specifies whether the control displays the Size, Type, Modified for folders objects.
Enabled	Enables or disables the control.
	Resumes painting the control after painting is suspended

EndUpdate	by the BeginUpdate method.
EventParam	Retrieves or sets a value that indicates the current's event parameter.
ExcludeFilter	Specifies the pattern used to exclude files from the control's list, like '*.tmp *.log'.
ExcludeFolderFilter	Retrieves or sets a value that indicates the folders being excluded.
ExecuteContextCommand	Executes a context menu command.
ExecuteContextMenu	Executes a command from the object's context menu.
ExecuteTemplate	Executes a template and returns the result.
Expand	Expands and selects a folder giving its path.
ExpandFolders	Retrieves or sets a value that indicates whether the control expands the folder objects.
ExpandOnDblClk	Retrieves or sets a value that indicates whether a folder is expanded by double click.
ExploreFromHere	Specifies the root folder for the control.
FileFromPoint	Retrieves the file from the point.
FileTypes	Retrieves the control's FileTypes collection.
FilterBarBackColor	Specifies the background color of the control's filter bar.
FilterBarCaption	Specifies the filter bar's caption.
FilterBarDropDownHeight	Specifies the height of the drop down filter window proportionally with the height of the control's list.
FilterBarDropDownWidth	Specifies the width of the drop down filter window proportionally with the width of the control's column.
FilterBarFont	Retrieves or sets the font for control's filter bar.
FilterBarForeColor	Specifies the foreground color of the control's filter bar.
FilterBarHeight	Specifies the height of the control's filter bar. If the value is less than 0, the filterbar is automatically resized to fit its description.
FilterBarPrompt	Specifies the caption to be displayed when the filter pattern is missing.
FilterBarPromptColumns	Specifies the list of columns to be used when filtering using the prompt.
FilterBarPromptPattern	Specifies the pattern for the filter prompt.

FilterBarPromptType	Specifies the type of the filter prompt.
FilterBarPromptVisible	Shows or hides the filter prompt.
FilterInclude	Specifies the items being included after the user applies the filter.
Font	Retrieves or sets the Font object used to paint control.
ForeColor	Retrieves or sets the control's foreground color.
ForeColorHeader	Specifies the header's foreground color.
FormatABC	Formats the A,B,C values based on the giving expression and returns the result.
FreezeEvents	Prevents the control to fire any event.
FullRowSelect	Enables full-row selection in the control.
Get	Builds and gets the collection of File objects of the given type.
HasButtons	Adds a button to the left side of each parent item.
HasCheckBox	Specifies whether the control displays a check box for each item.
HasLines	Retrieves or sets a value that indicates whether the control links the child items to their parents.
HasLinesAtRoot	Retrieves or sets a value that indicates whether the control draws the lines that link the root items.
HeaderAppearance	Retrieves or sets a value that indicates the header's appearance.
HeaderHeight	Retrieves or sets a value indicating the control's header height.
HeaderVisible	Retrieves or sets a value that indicates whether the control's header bar is visible or hidden.
HideSelection	Returns a value that determines whether selected item appears highlighted when a control loses the focus.
HotBackColor	Retrieves or sets a value that indicates the hot-tracking background color.
HotForeColor	Retrieves or sets a value that indicates the hot-tracking foreground color.
hWnd	Retrieves the control's window handle.
ImageSize	Retrieves or sets the size of icons the control displays.

IncludeFiles	Retrieves or sets a value indicating whether the control includes the files to the list.
IncludeFilesInFolder	Retrieves or sets a value that indicates whether the control includes files when expanding a folder.
IncludeFilter	Specifies the pattern used to include files to the control's list, like '*.cpp *.h'
IncludeFolderFilter	Retrieves or sets a value that indicates the folders being included.
IncludeFolders	Retrieves or sets a value that indicates whether the control includes the folders.
IncludeParent	Retrieves or sets a value that indicates whether the control includes the parent folder.
IncludeParentIconKey	Retrieves or sets a value that indicates the key of the icon used for 'Parent' button. Use LoadIcon property to load icons to control.
IncludeParentLabel	Specifies the label for the parent item.
IncludeSubFolderIconKey	Retrieves or sets a value that indicates the key of the icon to highlights folders that includes sub-folders.
IncrementalSearch	Specifies how the control searches for the objects while user types characters.
Indent	Retrieves or sets the amount, in pixels, that child items are indented relative to their parent items.
IsBusy	Indicates whether the control still collects information about current files and folders.
Layout	Saves or loads the control's layout, such as positions of the columns, scroll position, filtering values.
LoadIcon	Appends a new icon image to control images collection.
LoadIcons	Loads new images to control.
LoadIconsKey	Specifies the starting key when the LoadIcons method is used.
Loading	Specifies the HTML caption being displayed in the list if loading files or folders could take long time.
ModifiedDaysAgo	Specifies a value that indicates whether the Modified column shows the number of days ago when the file is last updated.
OLEDrag	Causes a component to initiate an OLE drag/drop

	operation.
OLEDropMode	Returns or sets how a target component handles drop operations
Option	Retrieves or sets a value that indicates an option for the control.
Picture	Retrieves or sets a graphic to be displayed in the control.
PictureDisplay	Retrieves or sets a value that indicates the way how the graphic is displayed on the control's background
Refresh	Refreshes the control.
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.
Search	Specifies the list of files and folders including wild card characters to search for.
SelBackColor	Retrieves or sets a value that indicates the selection background color.
Select	Selects a folder, giving its displaying name, relative or absolute path.
SelectByDrag	Specifies whether the user can select multiple files/folders by dragging.

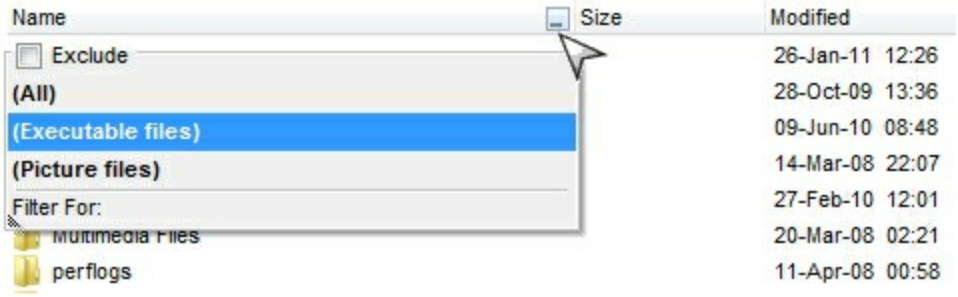
SelectOnRelease	Indicates whether the selection occurs when the user releases the mouse button.
SelForeColor	Retrieves or sets a value that indicates the selection foreground color.
ShowContextMenu	Specifies the object's context menu.
ShowFocusRect	Retrieves or sets a value indicating whether the control draws a thin rectangle arround the focused item.
SingleSel	Retrieves or sets a value indicating whether control support single or multiples selection.
Sort	Sorts a column.
Statistics	Gives statistics data of objects being hold by the control.
StopSearch	Stops the searching operation.
Template	Specifies the control's template.
TemplateDef	Defines inside variables for the next Template/ExecuteTemplate call.
TemplatePut	Defines inside variables for the next Template/ExecuteTemplate call.
ToolTipDelay	Specifies the time in ms that passes before the ToolTip appears.
ToolTipFont	Retrieves or sets the tooltip's font.
ToolTipPopDelay	Specifies the period in ms of time the ToolTip remains visible if the mouse pointer is stationary within a control.
ToolTipWidth	Specifies a value that indicates the width of the tooltip window, in pixels.
UseVisualTheme	Specifies whether the control uses the current visual theme to display certain UI parts.
Version	Retrieves the control's version.
VisualAppearance	Retrieves the control's appearance.

method ExFileView.AddColumnCustomFilter (ColumnName as String, Caption as String, Filter as String)

Adds a custom filter to the column. /*not supported in the lite version*/

Type	Description
ColumnName as String	A string expression that indicates the column's name. The valid values are: 'Name', 'Size', 'Modified', 'Type', 'In Folder'
Caption as String	A string expression that indicates the caption for filter pattern.
Filter as String	A string expression that indicates filter pattern being added. The Filter parameter supports wild characters like: ?, *, ...

Use the AddColumnCustomFilter method to add custom filter patterns to the control's drop down filter window. Use the ColumnFilterButton property to display the filter button in the column's header. Use the ColumnFilter, ColumnFilterType properties and ApplyFilter method to apply a filter to the control's content. Use the ClearColumnCustomFilters method to clear the list of custom filter patterns. Use the FilterBarDropDownHeight property to specify the drop down filter window. A pattern filter may contain the wild card characters **'?'** for any single character, **'*'** for zero or more occurrences of any character, '#' for any digit character, **'^'** negates the pattern, **'|'** determines the options in the pattern. For instance: '*.bat|*.exe' specifies all files that have the 'bat' or 'exe' extension.



The following VB sample adds custom filter patterns for executable files:

```
With ExFileView1
   .FilterBarDropDownHeight = 0.7
   .ColumnFilterButton("Name") = True
   .AddColumnCustomFilter "Name", "(Executable files)", "*.exe|*.com|*.bat"
End With
```

The following C++ sample adds custom filter patterns for executable files:

```
m_fileview.SetFilterBarDropDownHeight( 0.7 );
m_fileview.SetColumnFilterButton("Name", TRUE );
m_fileview.AddColumnCustomFilter("Name", "(Executable files)", "*.exe|*.com|*.bat");
```

The following VB.NET sample adds custom filter patterns for executable files:

```
With AxExFileView1
   .FilterBarDropDownHeight = 0.7
   .set_ColumnFilterButton("Name", True)
   .AddColumnCustomFilter("Name", "(Executable files)", "*.exe|*.com|*.bat")
End With
```

The following C# sample adds custom filter patterns for executable files:

```
axExFileView1.FilterBarDropDownHeight = 0.7;
axExFileView1.set_ColumnFilterButton("Name", true);
axExFileView1.AddColumnCustomFilter("Name", "(Executable files)", "*.exe|*.com|*.bat");
```

The following VFP sample adds custom filter patterns for executable files:

```
With thisform.ExFileView1
   .FilterBarDropDownHeight = 0.7
   .Object.ColumnFilterButton("Name") = .t.
   .AddColumnCustomFilter("Name", "(Executable files)", "*.exe|*.com|*.bat")
EndWith
```

property ExFileView.AllowEnterFolder as Boolean

Specifies whether a new folder is opened, once the user presses the Enter key or double-clicks a folder.

Type	Description
Boolean	A Boolean expression that specifies whether a new folder is opened, once the user presses the Enter key or double-clicks a folder.

By default, the AllowEnterFolder property is True. The AllowEnterFolder property specifies whether a new folder is opened, once the user presses the Enter key or double-clicks a folder.

property ExFileView.AllowMenuContext 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 AllowMenuContext 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. Use the Get property to retrieve the selected item(s). Use the Name property to specify the name of the file or the folder. Use the Folder property to specify whether the File object refers a file or a folder. 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.

property ExFileView.AllowRename as Boolean

Retrieves or sets a value that indicates whether the control allows renaming items.

Type	Description
Boolean	A boolean expression that indicates whether the control allows renaming items.

By default, the AllowRename property is False. The AllowRename property specifies whether the control allows renaming files or folders. If the AllowRename property is True, a file/folder can be renamed at runtime, by pressing F2 or by pressing twice the left mouse button. Use the AllowContextMenu property to specify whether the control displays the file/folder's context menu when user does a right click. Use the ExecuteContextCommand method to execute a command from the file's context menu.

property ExFileView.AllowSelectNothing as Boolean

Specifies whether the current selection is erased, once the user clicks outside of the items section.

Type	Description
Boolean	A Boolean expression that specifies whether the current selection is erased, once the user clicks outside of the items section.

By default, the AllowSelectNothing property is False. The AllowSelectNothing property specifies whether the current selection is erased, once the user clicks outside of the items section. For instance, if the control's SingleSel property is True, and AllowSelectNothing property is True, you can un-select the single-selected item if pressing the CTRL + Space, or by CTRL + click.

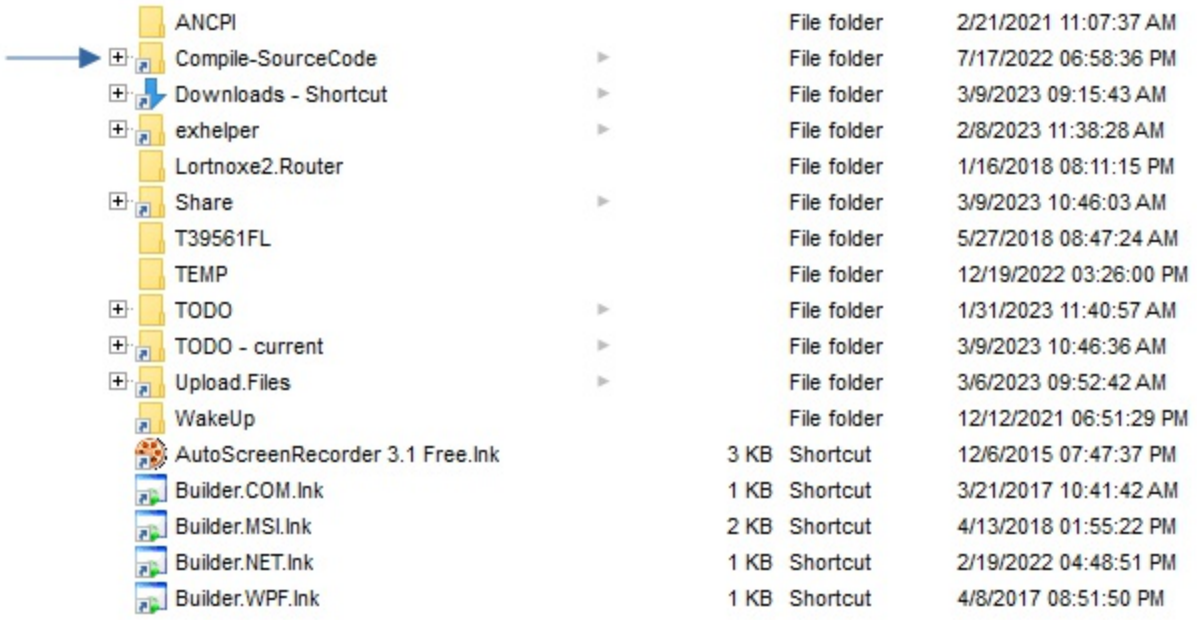
property ExFileView.AllowShortcutFolders as Boolean

Specifies whether the shortcut-folders are shown as folders or files.

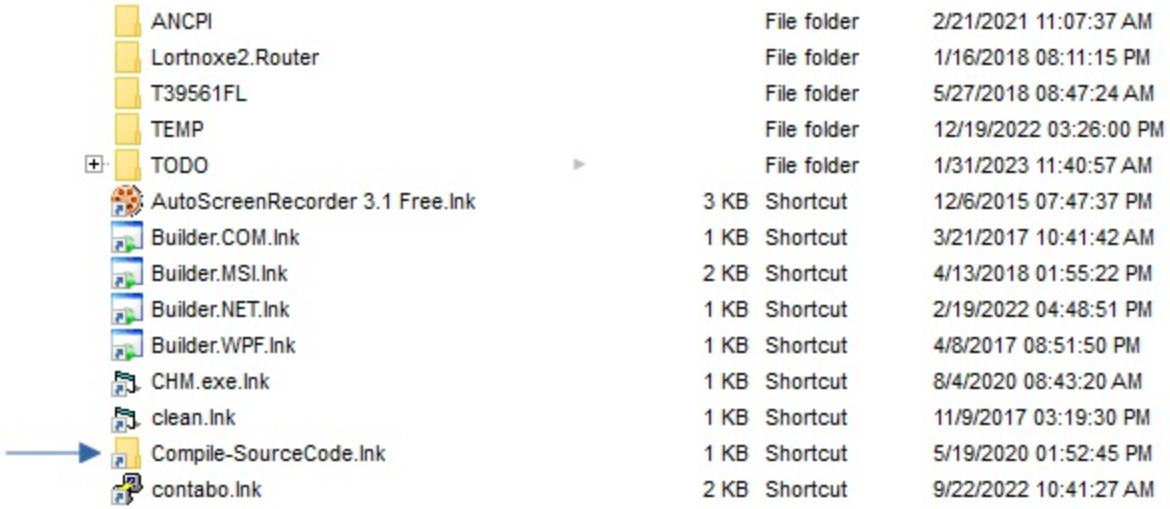
Type	Description
Boolean	A boolean expression that specifies whether the shortcut-folders are shown as folders or files

By default, the AllowShortcutFolders property is True. Use the AllowShortcutFolders property on False to show shortcut-folders as files. The ExpandFolders property retrieves or sets a value that indicates whether the control expands the folder objects. Use the ExploreFromHere property specifies the root folder for the control.

The following screen shot shows the shortcut folders as folders (default):



The following screen shot shows the shortcut folders as files:

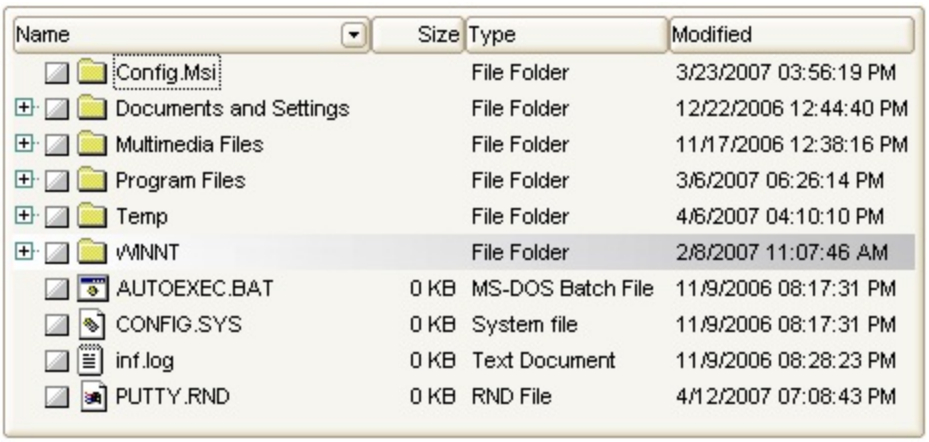


property ExFileView.Appearance as AppearanceEnum

Retrieves or sets the control's appearance.

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

Use the Appearance property to specify the control's border. Use the HeaderAppearance property to change the control's header bar appearance. Use the Add method to add new skins to the control. Use the BackColor property to specify the control's background color.



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

```
With ExFileView1
   .BeginUpdate
      .VisualAppearance.Add &H16, "c:\temp\frame.ebn"
      .Appearance = &H16000000
      .BackColor = RGB(250, 250, 250)
```

```
    .EndUpdate
End With
```

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

```
With AxExFileView1
    .BeginUpdate()
    .VisualAppearance.Add(&H16, "c:\temp\frame.ebn")
    .Appearance = &H16000000
    .BackColor = Color.FromArgb(250, 250, 250)
    .EndUpdate()
End With
```

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

```
axExFileView1.BeginUpdate();
axExFileView1.VisualAppearance.Add(0x16, "c:\\temp\\frame.ebn");
axExFileView1.Appearance = (EXFILEVIEWLib.AppearanceEnum)0x16000000;
axExFileView1.BackColor = Color.FromArgb(250, 250, 250);
axExFileView1.EndUpdate();
```

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

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

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

```
with thisform.ExFileView1
    .BeginUpdate
        .VisualAppearance.Add(0x16, "c:\temp\frame.ebn")
        .Appearance = 0x16000000
        .BackColor = RGB(250, 250, 250)
    .EndUpdate
endwith
```

method ExFileView.ApplyFilter ()

Applies the filter. /*not supported in the lite version*/

Type	Description

The ApplyFilter method updates the control's content once that user sets the filter using the ColumnFilter and ColumnFilterType properties. Use the ClearFilter method to clear the control's filter. Use the ColumnFilterButton property to show the filter drop down button in the column's caption.

The following VB sample filters for executable files:

```
With ExFileView1
   .ColumnFilterButton("Name") = True
   .ColumnFilter("Name") = "*.exe|*.com|*.bat"
   .ColumnFilterType("Name") = exPattern
   .ApplyFilter
End With
```

The following C++ sample filters for executable files:

```
m_fileview.SetColumnFilterButton("Name", TRUE );
m_fileview.SetColumnFilter( "Name", "*.exe|*.com|*.bat" );
m_fileview.SetColumnFilterType( "Name", 1 /*exPattern*/ );
m_fileview.ApplyFilter();
```

The following VB.NET sample filters for executable files:

```
With AxExFileView1
   .set_ColumnFilterButton("Name", True)
   .set_ColumnFilter("Name", "*.exe|*.com|*.bat")
   .set_ColumnFilterType("Name", EXFILEVIEWLib.FilterTypeEnum.exPattern)
   .ApplyFilter()
End With
```

The following C# sample filters for executable files:

```
axExFileView1.set_ColumnFilterButton("Name", true);
axExFileView1.set_ColumnFilter("Name", "*.exe|*.com|*.bat");
axExFileView1.set_ColumnFilterType("Name", EXFILEVIEWLib.FilterTypeEnum.exPattern);
```

```
axExFileView1.ApplyFilter();
```

The following VFP sample filters for executable files:

```
With thisform.FileView1
   .Object.ColumnFilterButton("Name") = .t.
   .Object.ColumnFilter("Name") = "*.exe|*.com|*.bat"
   .Object.ColumnFilterType("Name") = 1
   .ApplyFilter
EndWith
```

property ExFileView.Asynchronous as Boolean

Specifies whether the files and folders information is loading in the background.

Type	Description
Boolean	A boolean expression that specifies whether the control loads files/folder in the background.

By default, the Asynchronous property is False. The Asynchronous property on True, improves drastically the performance while loading ten of thousand of files/folders. The StateChange(BusyState) event notifies the application once the control is busy updating data to view. The StateChange(ReadyState) event notifies the application once the control done loading the files/folders into the view.

method ExFileView.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 FileView1_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.

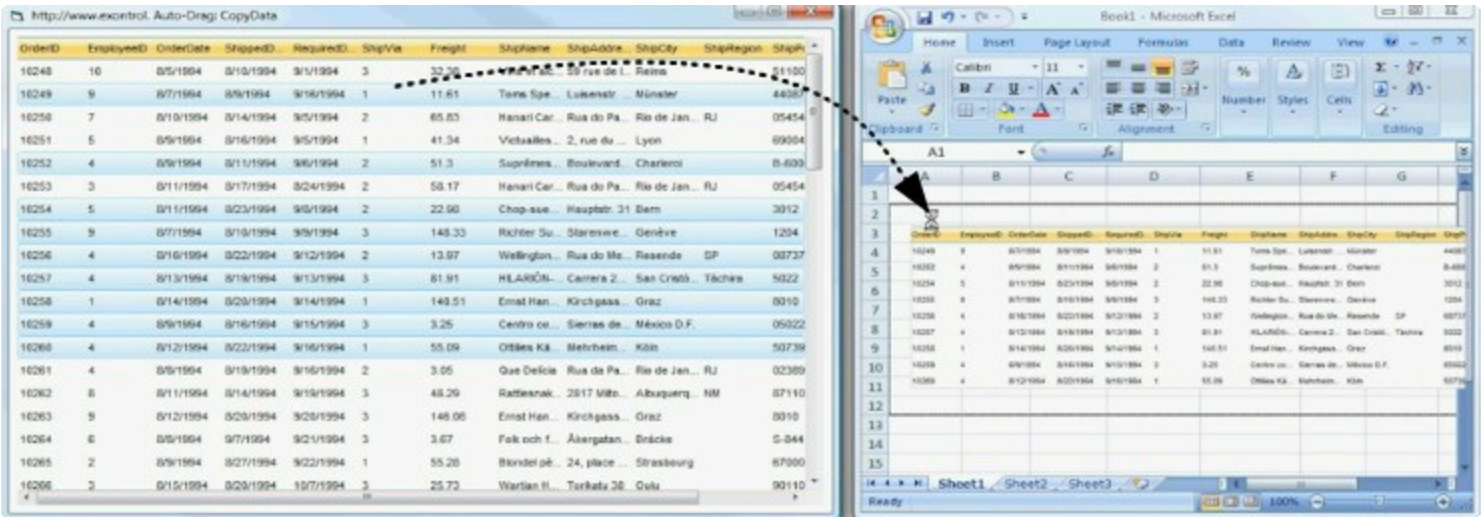
The advantage of the AttachTemplate relative to Template / ExecuteTemplate is that the AttachTemplate can add handlers to the control events.

property ExFileView.AutoDrag as AutoDragEnum

Gets or sets a value that indicates the way the component supports the AutoDrag feature.

Type	Description
AutoDragEnum	An AutoDragEnum expression that specifies what the control does once the user clicks and start dragging an item.

By default, the AutoDrag property is exAutoDragNone(0). The AutoDrag feature indicates what the control does when the user clicks an item and starts dragging it. For instance, using the AutoDrag feature you can automatically lets the user to drag and drop the data to OLE compliant applications like Microsoft Word, Excel and so on. The SingleSel property specifies whether the control supports single or multiple selection.



Use the AutoDrag property to allow Drag and Drop operations like follows:

- Ability to drag and drop the data as *text*, to your favorite Office applications, like Word, Excel, or any other OLE-Automation compliant
- Ability to drag and drop the data as it *looks*, to your favorite Office applications, like Word, Excel, or any other OLE-Automation compliant
- Ability to smoothly scroll the control's content moving the mouse cursor up or down
- and more ...

property ExFileView.AutoUpdate as Boolean

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

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

Determines if the control auto updates files/folders if the user changes the folder structure in the background. If the AutoUpdate property is True, the control receives notification messages each time the folder structure is changed - for example, in another Explorer window that is running. If the user or another program changes the folder structure , the control will update itself accordingly. If the AutoUpdate property is False, the changes in the folder will be updated next time when the control is refreshed. Call the Refresh method to update the control's content. The control has tow different ways of updating the current list: by refreshing, or by applying the changes only. If the ChangeNotification property is True, the control applies only the changes in the browsed folder. Use the BrowseFolderPath property to specify the path of the browsed folder.

property ExFileView.BackColor as Color

Retrieves or sets the control's background.

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

Use the BackColor property to specify the control's background color. Use the ForeColor property to change the control's foreground color. Use the BackColor property to specify the background color for files or folders that match specified patterns. Use the BackColorHeader property to specify the background color for the control's header bar. Use the FilterBarBackColor property to specify the background color for the control's filter bar. Use the SelForeColor and SelBackColor properties to specify the background and foreground colors for selected items.

property ExFileView.BackColorHeader as Color

Specifies the header's background color.

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

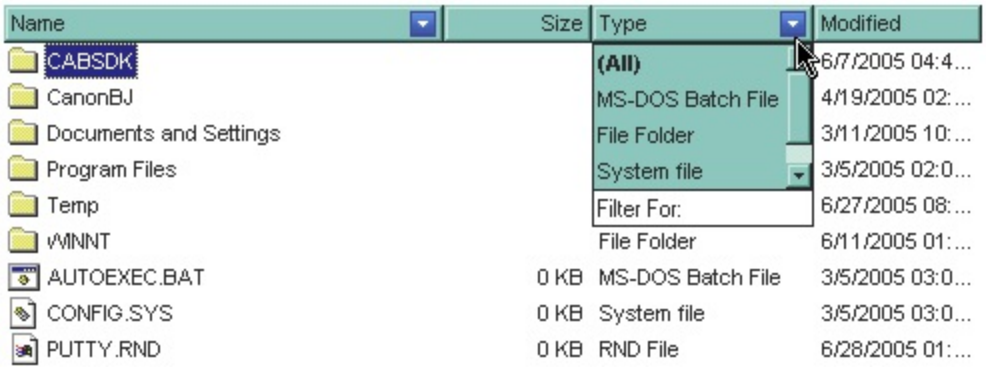
Use the BackColorHeader property to specify the background color for the control's header bar. Use the ForeColorHeader properties to specify the foreground color for the control's header bar. The HeaderVisible property shows or hides the control's header bar. Use the FilterBarBackColor property to specify the background color for the control's filter bar. Use the BackColor property to specify the control's background color. Use the SelForeColor and SelBackColor properties to specify the background and foreground colors for selected items. Use the Background(exCursorHoverColumn) property to specify the column's appearance when the cursor hovers it.

property ExFileView.Background(Part as BackgroundPartEnum) as Color

Returns or sets a value that indicates the background color for parts in the control. /*not supported in the lite version*/

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

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



The following VB sample changes the visual appearance for the "drop down" filter button. The sample applies the skin "" to the "drop down" filter buttons:

```
With ExFileView1
   With .VisualAppearance
      .Add &H1, App.Path + "\fbardd.ebn"
   End With
   .Background(exHeaderFilterBarButton) = &H1000000
End With
```

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

```
#include "Appearance.h"
m_fileview.GetVisualAppearance().Add( 0x01,
COleVariant(_T("D:\\Temp\\ExFileView.Help\\fbardd.ebn")) );
m_fileview.SetBackground( 0 /*exHeaderFilterBarButton*/, 0x1000000 );
```

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

```
With AxExFileView1
   With .VisualAppearance
      .Add(&H1, "D:\Temp\ExFileView.Help\fbardd.ebn")
   End With
   .set_Background(EXFILEVIEWLib.BackgroundPartEnum.exHeaderFilterBarButton,
&H1000000)
End With
```

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

```
axExFileView1.VisualAppearance.Add(0x1, "D:\\Temp\\ExFileView.Help\\fbardd.ebn");
axExFileView1.set_Background(EXFILEVIEWLib.BackgroundPartEnum.exHeaderFilterBarButto
 0x1000000);
```

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

```
With thisform.ExFileView1
   With .VisualAppearance
      .Add(1, "D:\Temp\ExFileView.Help\fbardd.ebn")
   EndWith
   .Object.Background(0) = 16777216
EndWith
```

The 16777216 value is the 0x1000000 value in hexadecimal.

method ExFileView.BeginUpdate ()

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

Type	Description

property ExFileView.BrowseFolderPath as String

Retrieves or sets the browsed folder path.

Type	Description
String	A string expression that defines the browsed folder path.

Use the BrowseFolderPath to change the browsed folder. Call the ExploreFromHere property to set the folder that's the root of the control. The ExploreFromHere property sets also the BrowseFolderPath to the newly root folder. Use "" (empty string) to browse "My computer" folder. *The BrowseFolderPath property has no effect (returns empty string), if the control's ExploreFromHere property includes |, > or \r\n characters (shortly the BrowseFolderPath property has no effect if the control display multiple root-folders).*

The control fires the StateChange event when the user changes the browsed path. The images collection is updated each time a new folder is browsed. Use the LoadIcon or LoadIcons method to add new icons to the control. Use the AutoUpdate and ChangeNotification properties to update the control's content when changes occur in the browsed folder. Use the Expand method to programmatically expand a folder giving its path.

If changing the folder fails, the BrowseFolderPath property fires the exception "The BrowseFolderPath property isn't a valid path or the path is not derived from ExploreFromHere.". In order to prevent the exception you can *on error* handler like in the following samples:

```
On Error Resume Next
With ExFileView1
   .ExploreFromHere = "\\tsclient"
   .BrowseFolderPath = "D\Users\Mihai"
End With
```

or in VFP:

```
ON ERROR return
with thisform.exFileView1
   .ExploreFromHere = "\\tsclient"
   .BrowseFolderPath = "D\Users\Mihai"
endwith
```

The following VB sample changes the browsed folder:

```
With ExFileView1
   .BrowseFolderPath = "C:\Temp"
End With
```

The following C++ sample changes the browsed folder:

```
m_fileview.SetBrowseFolderPath( "C:\\Temp" );
```

The following VB.NET sample changes the browsed folder:

```
With AxExFileView1
   .BrowseFolderPath = "c:\temp"
End With
```

The following C# sample changes the browsed folder:

```
axExFileView1.BrowseFolderPath = "c:\\temp";
```

The following VFP sample changes the browsed folder:

```
With thisform.ExFileView1
   .BrowseFolderPath = "c:\temp"
EndWith
```

property ExFileView.ChangeNotification as Boolean

Enables or disables control's notifications by firing Change event, whether the control's list was altered.

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

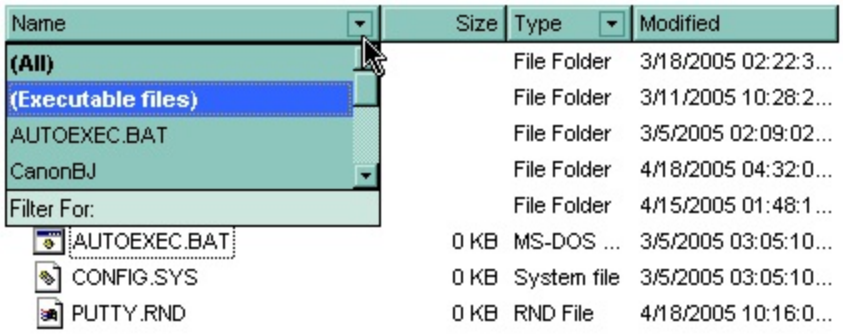
The ChangeNotification has effect only if the AutoUpdate property is True. If the ChangeNotification property is True, the control fires Change event each time a new change occurs in the browsed folder (adding, removing, changing files, or folders). Use the State property to determine the new state of the file or folder. Use the Folder property to specify whether the object holds information about a folder of a file. Use the BrowseFolderPath property to indicates the browsed folder.

method ExFileView.ClearColumnCustomFilters (ColumnName as String)

Clears the list of column's custom filters. /*not supported in the lite version*/

Type	Description
ColumnName as String	A string expression that indicates the column's name. The valid values are: 'Name', 'Size', 'Modified', 'Type', 'In Folder'

Use the ClearColumnCustomFilters method to clear the column's list of filter patterns added using the AddColumnCustomFilter method. By default, the control adds no custom filter patterns. Use the ClearFilter method to remove the control's filter.

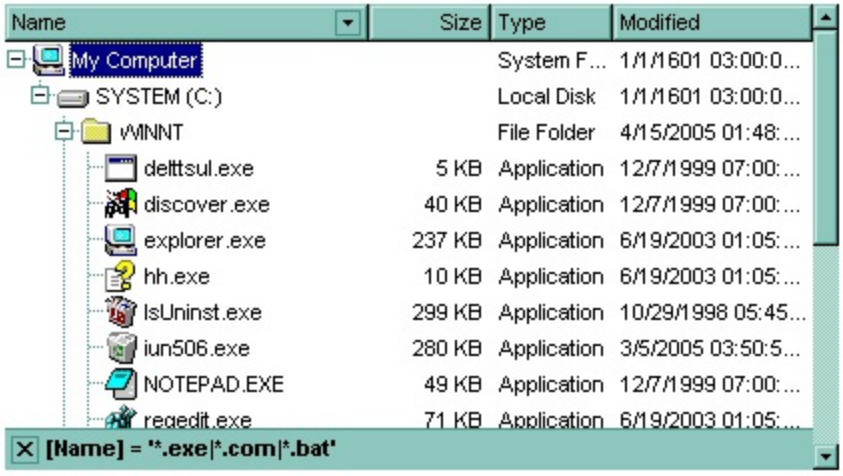


method ExFileView.ClearFilter ()

Clears the filter. /*not supported in the lite version*/

Type	Description

The method clears the ColumnFilter and ColumnFilterType properties for all columns in the control. The ApplyFilter method is automatically called when ClearFilter method is invoked. Use the FilterBarHeight property to hide the control's filter bar. Use the FilterBarCaption property to specify the caption in the control's filter bar. Use the Description property to change predefined strings in the control's filter bar.



method ExFileView.ClearImages ()

Clears the loaded images.

Type	Description

Clears the images previously loaded using the LoadIcon or LoadIcons methods. Use the Refresh method to update the control's content. Use the IconIndex property of FileType object to change the file's icon. Use the BrowseFolderPath property to specify the path to the browsed folder. The control fires the StateChange event when the user changes the browsed path.

property ExFileView.ColumnAutoResize as Boolean

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

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

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

property ExFileView.ColumnCaption(ColumnName as String) as String

Specifies the column's caption.

Type	Description
ColumnName as String	A string expression that indicates the column's name. The valid values are: 'Name', 'Size', 'Modified', 'Type', 'In Folder'.
String	A string expression that indicates the column's caption using built-in HTML tags.

Use the ColumnCaption property to specify the column's caption. The column's caption is displayed on the control's header bar. The HeaderVisible property specifies whether the control displays its header bar. Use the ColumnVisible property to hide or show a column. Use the BackColorHeader property to specify the background color for the control's header bar. Use the ForeColorHeader property to specify the foreground color for the control's header bar.

In VFP environment, you need to call **Object** property of the control before calling the ColumnCaption property, else an error occurs.

The ColumnCaption property supports 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:

outlined

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

shadow

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

outline anti-aliasing

The following VB sample changes the caption of the 'Name' column':

```
ExFileView1.ColumnCaption("Name") = "Verzeichnis"
```

The following C++ sample changes the caption of the 'Name' column':

```
m_fileview.SetColumnCaption( "Name", "Verzeichnis" );
```

The following VB.NET sample changes the caption of the 'Name' column':

```
AxExFileView1.set_ColumnCaption("Name", "Verzeichnis")
```

The following C# sample changes the caption of the 'Name' column':

```
axExFileView1.set_ColumnCaption("Name", "Verzeichnis");
```

The following VFP sample changes the caption of the 'Name' column:

```
with thisform.FileView1
  .Object.ColumnCaption("Name") = "Verzeichnis"
endwith
```

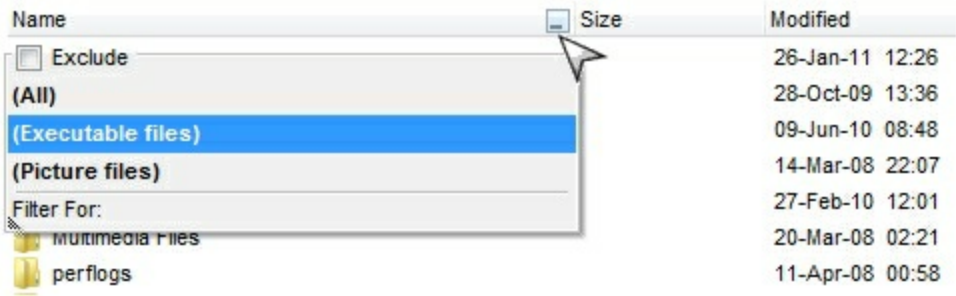
property ExFileView.ColumnFilter(ColumnName as String) as String

Specifies the column's filter when filter type is exFilter or exPattern. /*not supported in the lite version*/

Type	Description
ColumnName as String	A string expression that indicates the column's name. The valid values are: 'Name', 'Size', 'Modified', 'Type', 'In Folder'
String	A string expression that specifies the column's filter.

If the ColumnFilterType property is exFilter the ColumnFilter property specifies the list of values used in filtering. The values are separated by '|' character. For instance if the Filter property is "CellA| CellB" the control includes only the items that have captions like: "CellA" or "CellB". If the ColumnFilterType is exPattern the ColumnFilter property defines the list of patterns used in filtering. The patterns are separator by '|' character. A pattern filter may contain the wild card characters '?' for any single character, '*' for zero or more occurrences of any character, '#' for any digit character, '^' negates the pattern, '|' separates the options in the pattern. For instance: '1*|2*' specifies all items that start with '1' or '2'. Use the FilterBarCaption property to change the caption for control's filter bar header. Use the AddColumnCustomFilter method to add custom filters to the column.

The ApplyFilter method should be called to update the control's content after changing the Filter or FilterType property. The ClearFilter method clears the control's filter.



The following VB sample filters for executable files:

```
With ExFileView1
   .ColumnFilterButton("Name") = True
   .ColumnFilter("Name") = "*.exe|*.com|*.bat"
   .ColumnFilterType("Name") = exPattern
   .ApplyFilter
End With
```

The following C++ sample filters for executable files:

```
m_fileview.SetColumnFilterButton("Name", TRUE );
m_fileview.SetColumnFilter( "Name", "*.exe|*.com|*.bat" );
m_fileview.SetColumnFilterType( "Name", 1 /*exPattern*/ );
m_fileview.ApplyFilter();
```

The following VB.NET sample filters for executable files:

```
With AxExFileView1
   .set_ColumnFilterButton("Name", True)
   .set_ColumnFilter("Name", "*.exe|*.com|*.bat")
   .set_ColumnFilterType("Name", EXFILEVIEWLib.FilterTypeEnum.exPattern)
   .ApplyFilter()
End With
```

The following C# sample filters for executable files:

```
axExFileView1.set_ColumnFilterButton("Name", true);
axExFileView1.set_ColumnFilter("Name", "*.exe|*.com|*.bat");
axExFileView1.set_ColumnFilterType("Name", EXFILEVIEWLib.FilterTypeEnum.exPattern);
axExFileView1.ApplyFilter();
```

The following VFP sample filters for executable files:

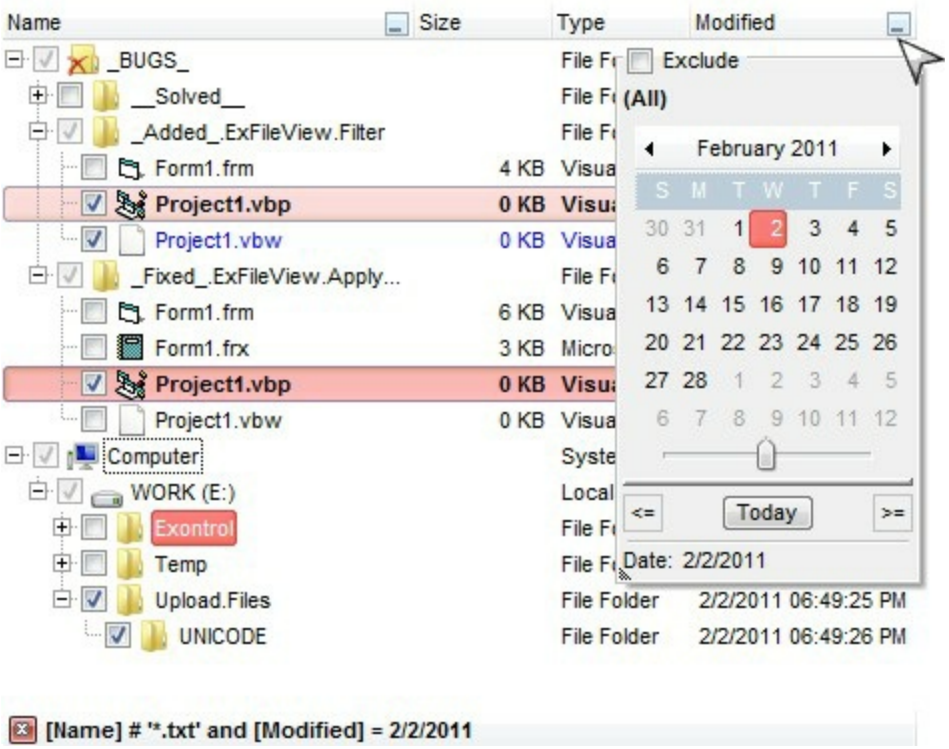
```
With thisform.FileView1
   .Object.ColumnFilterButton("Name") = .t.
   .Object.ColumnFilter("Name") = "*.exe|*.com|*.bat"
   .Object.ColumnFilterType("Name") = 1
   .ApplyFilter
EndWith
```

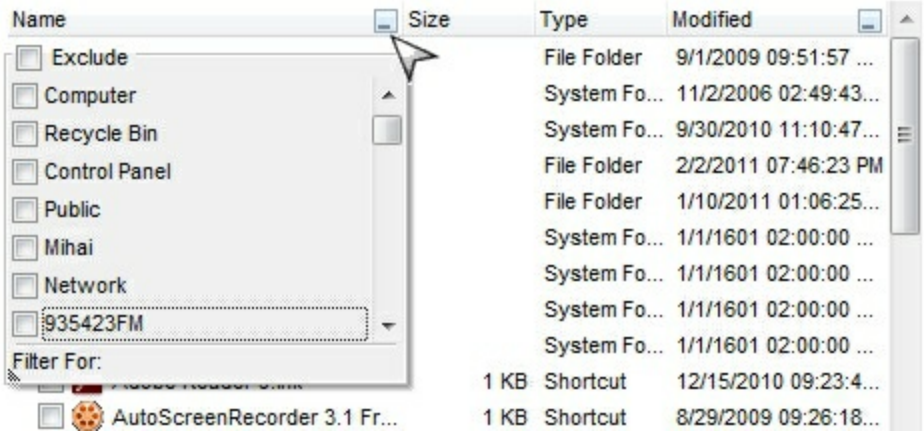
property ExFileView.ColumnFilterButton(ColumnName as String) as Boolean

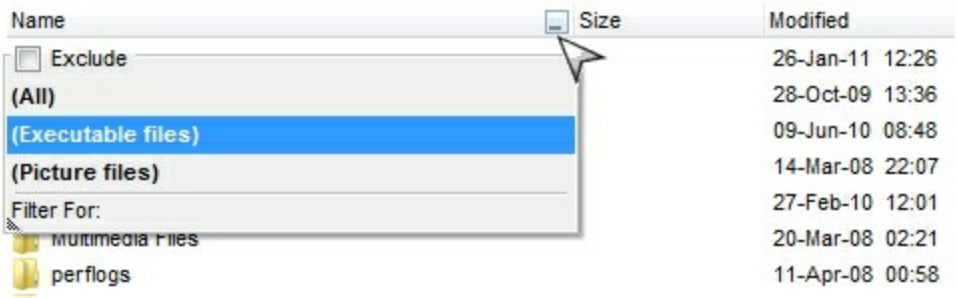
Specifies a value that indicates whether the column displays the filter button. /*not supported in the lite version*/

Type	Description
ColumnName as String	A string expression that indicates the column's name. The valid values are: 'Name', 'Size', 'Modified', 'Type', 'In Folder'. The Modified column displays a calendar control that can be used to filter for files/folder modified within a range and so.
Boolean	A Boolean expression that indicates whether the column displays the filter button.

Use the ColumnFilterButton property to show or hide the column's filter button. By default, the control displays no filter buttons. Use the AddColumnCustomFilter method to add custom filter patterns to the column. Use the FilterBarDropDownHeight property to specify the height of the control's drop down filter window. Use the FilterBarDropDownWidth property to specify the width of the control's drop down filter window. Use the exHideFileExtensionsForKnownFileTypes option to show the file extensions in case your Windows Explorer, the "**Hide File Extensions For Known File Types**" is checked. Use the FilterInclude property to specify whether the child files or folders are included in the list, after user applies the filter.







The following VB sample adds custom filter patterns for executable files:

```
With ExFileView1
   .FilterBarDropDownHeight = 0.7
   .ColumnFilterButton("Name") = True
   .AddColumnCustomFilter "Name", "(Executable files)", "*.exe|*.com|*.bat"
End With
```

The following C++ sample adds custom filter patterns for executable files:

```
m_fileview.SetFilterBarDropDownHeight( 0.7 );
m_fileview.SetColumnFilterButton("Name", TRUE );
m_fileview.AddColumnCustomFilter("Name", "(Executable files)", "*.exe|*.com|*.bat");
```

The following VB.NET sample adds custom filter patterns for executable files:

```
With AxExFileView1
   .FilterBarDropDownHeight = 0.7
   .set_ColumnFilterButton("Name", True)
   .AddColumnCustomFilter("Name", "(Executable files)", "*.exe|*.com|*.bat")
End With
```

The following C# sample adds custom filter patterns for executable files:

```
axExFileView1.FilterBarDropDownHeight = 0.7;
axExFileView1.set_ColumnFilterButton("Name", true);
axExFileView1.AddColumnCustomFilter("Name", "(Executable files)", "*.exe|*.com|*.bat");
```

The following VFP sample adds custom filter patterns for executable files:

```
With thisform.ExFileView1
   .FilterBarDropDownHeight = 0.7
   .Object.ColumnFilterButton("Name") = .t.
   .AddColumnCustomFilter("Name", "(Executable files)", "*.exe|*.com|*.bat")
EndWith
```

property ExFileView.ColumnFilterType(ColumnName as String) as FilterTypeEnum

Specifies the column's filter type. /*not supported in the lite version*/

Type	Description
ColumnName as String	A string expression that indicates the column's name. The valid values are: 'Name', 'Size', 'Modified', 'Type', 'In Folder'
FilterTypeEnum	A FilterTypeEnum expression that indicates the column's filter type.

The ColumnFilterType property defines the filter's type on specified column. By default, the ColumnFilterType property is exAll. No filter is applied if the ColumnFilterType is exAll. The ColumnFilter property defines the column's filter. Use the ColumnFilterButton property to display the column's filter button. Use the FilterBarCaption property to specify the caption for the control's filter bar header.

The ApplyFilter method should be called to update the control's content after changing the ColumnFilter or ColumnFilterType property. The ClearFilter method clears the control's filter.

The following VB sample filters for executable files:

```
With ExFileView1
   .ColumnFilterButton("Name") = True
   .ColumnFilter("Name") = "*.exe|*.com|*.bat"
   .ColumnFilterType("Name") = exPattern
   .ApplyFilter
End With
```

The following C++ sample filters for executable files:

```
m_fileview.SetColumnFilterButton("Name", TRUE );
m_fileview.SetColumnFilter( "Name", "*.exe|*.com|*.bat" );
m_fileview.SetColumnFilterType( "Name", 1 /*exPattern*/ );
m_fileview.ApplyFilter();
```

The following VB.NET sample filters for executable files:

```
With AxExFileView1
   .set_ColumnFilterButton("Name", True)
```

```
    .set_ColumnFilter("Name", "*.exe|*.com|*.bat")
    .set_ColumnFilterType("Name", EXFILEVIEWLib.FilterTypeEnum.exPattern)
    .ApplyFilter()
End With
```

The following C# sample filters for executable files:

```
axExFileView1.set_ColumnFilterButton("Name", true);
axExFileView1.set_ColumnFilter("Name", "*.exe|*.com|*.bat");
axExFileView1.set_ColumnFilterType("Name", EXFILEVIEWLib.FilterTypeEnum.exPattern);
axExFileView1.ApplyFilter();
```

The following VFP sample filters for executable files:

```
With thisform.FileView1
    .Object.ColumnFilterButton("Name") = .t.
    .Object.ColumnFilter("Name") = "*.exe|*.com|*.bat"
    .Object.ColumnFilterType("Name") = 1
    .ApplyFilter
EndWith
```

property ExFileView.ColumnsAllowSizing as Boolean

Retrieves or sets a value that indicates whether a user can resize columns at run-time.

Type	Description
Boolean	A Boolean expression that indicates whether a user can resize columns at run-time.

By default, the ColumnsAllowSizing property is False. The ColumnsAllowSizing is used to specify whether the user can resize the columns in the list section as well, not only on the header. If this is true, you can have the cursor between columns not necessary into the header also in the list section. If False, the user still can resize the columns using the header of the control. Use the HeaderVisible property to show or hide the control's header bar. The HeaderAppearance property specifies the appearance of the column in the control's header bar.

property ExFileView.ColumnsVisible as FileColumnEnum

Indicates the columns being visible.

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

By default, the ColumnsVisible property is exFileColumnName | exFileColumnSize | exFileColumnType | exFileColumnModified (30). The ColumnsVisible property indicates the columns being visible. You can use the ColumnsVisible property to show/hide multiple columns at once.

property ExFileView.ColumnVisible(ColumnName as String) as Boolean

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

Type	Description
ColumnName as String	A String expression that indicates the columns name. The following values are valid: 'Name', 'Size', 'Type'and 'Modified'.
Boolean	A boolean expression that indicates whether the column is visible or hidden.

Use the ColumnVisible property to hide a column. Use the ColumnWidth property to change the column's width. By default, the "Name", "Size", "Type" and "Modified" columns are visible. Use the HeaderVisible property to hide the control's header bar. Use the BackColorHeader property to specify the background color for the control's header bar. Use the ForeColorHeader property to specify the foreground color for the control's header bar. You can use the ColumnsVisible property to show/hide multiple columns at once.

The following VB sample hides the 'Size' column:

```
ExFileView1.ColumnVisible("Size") = False
```

The following C++ sample hides the 'Size' column:

```
m_fileview.SetColumnVisible( "Size", FALSE );
```

The following VB.NET sample hides the 'Size' column:

```
AxExFileView1.set_ColumnVisible("Size", False)
```

The following C# sample hides the 'Size' column:

```
axExFileView1.set_ColumnVisible("Name", False);
```

The following VFP sample hides the 'Size' column:

```
with thisform.FileView1
  .Object.ColumnVisible("Size") = .f.
endwith
```

property ExFileView.ColumnWidth(ColumnName as String) as Long

Retrieves or sets a value that indicates the column's width.

Type	Description
ColumnName as String	A String expression that indicates the columns nam. One of the following is valid: "Name", "Size", "Type" or "Modified"
Long	A long expression that indicates the column's width in pixels.

Use the ColumnWidth property to change the column's width. Use the ColumnVisible to hide a column. By default, all columns are visible. Use the ColumnVisible property to hide a column. Use the ColumnCaption property to specify the column's caption. Use the HeaderVisible property to show or hide the control's header bar. Use the ColumnAutoResize property to specify whether the control resizes the visible columns so all fit the control's client area.

The following VB sample changes the 'Size' column width:

```
ExFileView1.ColumnWidth("Size") = 16
```

The following C++ sample changes the 'Size' column width:

```
m_fileview.SetColumnWidth( "Size", 16 );
```

The following VB.NET sample changes the 'Size' column width:

```
AxExFileView1.set_ColumnWidth("Size", 16)
```

The following C# sample changes the 'Size' column width:

```
axExFileView1.set_ColumnWidth("Name", 16);
```

The following VFP sample changes the 'Size' column width:

```
with thisform.FileView1
  .Object.ColumnWidth("Size") = 16
endwith
```

method ExFileView.Copy ()

Copies the control's content to the clipboard, in the EMF format.

Type	Description

Use the Copy method to copy the control's content to the clipboard, in Enhanced Metafile (EMF) format. The Enhanced Metafile format is a 32-bit format that can contain both vector information and bitmap information. Use the CopyTo method to copy the control's content to EMF/BMP/GIF/PNG/JPEG or PDF files.

This format is an improvement over the Windows Metafile Format and contains extended features, such as the following:

Built-in scaling information
Built-in descriptions that are saved with the file
Improvements in color palettes and device independence

The EMF format is an extensible format, which means that a programmer can modify the original specification to add functionality or to meet specific needs. You can paste this format to Microsoft Word, Excel, Front Page, Microsoft Image Composer and any application that know to handle EMF formats.

The Copy method copies the control's header if it's visible, and all visible items. The items are not expanded, they are listed in the order as they are displayed on the screen. Use the HeaderVisible property to show or hide the control's header. The background of the copied control is transparent.

The following VB sample saves the control's content to a EMF file, when user presses the CTRL+C key:

```
Private Sub ExFileView1_KeyDown(KeyCode As Integer, Shift As Integer)
   If (KeyCode = vbKeyC) And Shift = 2 Then
      Clipboard.Clear
      ExFileView1.Copy
      SavePicture Clipboard.GetData(), App.Path & "\test.emf"
   End If
End Sub
```

Now, you can open your MS Windows Word application, and you can insert the file using the Insert\Picture\From File menu, or by pressing the CTRL+V key to paste the clipboard.

The following C++ function saves the clipboard's data (EMF format) to a picture file:

```
BOOL saveEMFtoFile( LPCTSTR szFileName )
{
   BOOL bResult = FALSE;
   if ( ::OpenClipboard( NULL ) )
   {
      CComPtr<IPicture> spPicture;
      PICTDESC pictDesc = {0};
      pictDesc.cbSizeofstruct = sizeof(pictDesc);
      pictDesc.emf.hemf = (HENHMETAFILE)GetClipboardData( CF_ENHMETAFILE );
      pictDesc.picType = PICTYPE_ENHMETAFILE;
      if ( SUCCEEDED( OleCreatePictureIndirect( &pictDesc, IID_IPicture, FALSE,
(LPVOID*)&spPicture ) ) )
      {
         HGLOBAL hGlobal = NULL;
         CComPtr<IStream> spStream;
         if ( SUCCEEDED( CreateStreamOnHGlobal( hGlobal = GlobalAlloc( GPTR, 0 ), TRUE,
&spStream ) ) )
         {
            long dwSize = NULL;
            if ( SUCCEEDED( spPicture->SaveAsFile( spStream, TRUE, &dwSize ) ) )
            {
               USES_CONVERSION;
               HANDLE hFile = CreateFile( szFileName, GENERIC_WRITE, NULL, NULL,
CREATE_ALWAYS, NULL, NULL );
               if ( hFile != INVALID_HANDLE_VALUE )
               {
                  LARGE_INTEGER l = {NULL};
                  spStream->Seek(l, STREAM_SEEK_SET, NULL);
                  long dwWritten = NULL;
                  while ( dwWritten < dwSize )
                  {
                     unsigned long dwRead = NULL;
                     BYTE b[10240] = {0};
                     spStream->Read( &b, 10240, &dwRead );
                     DWORD dwBWritten = NULL;
                     WriteFile( hFile, b, dwRead, &dwBWritten, NULL );
                     dwWritten += dwBWritten;
```

```
                }
                CloseHandle( hFile );
                bResult = TRUE;
            }
        }
    }
  }
  CloseClipboard();
}
return bResult;
}
```

The following VB.NET sample copies the control's content to the clipboard (open the mspaint application and paste the clipboard, after running the following code):

```
Clipboard.Clear()
With AxExFileView1
  .Copy()
End With
```

The following C# sample copies the control's content to a file (open the mspaint application and paste the clipboard, after running the following code):

```
Clipboard.Clear;
axExFileView1.Copy();
```

property ExFileView.CopyTo (File as String) as Variant

Exports the control's view to an EMF file.

Type	Description
File as String	A String expression that indicates the name of the file to be saved. If present, the CopyTo property retrieves True, if the operation succeeded, else False it is failed. If the File parameter is missing or empty, the CopyTo property retrieves an one dimension safe array of bytes that contains the EMF content. If the File parameter is not empty, the extension (characters after last dot) determines the graphical/ format of the file to be saved as follows: • ***.bmp *.dib *.rle**, saves the control's content in **BMP** format. • ***.jpg *.jpe *.jpeg *.jfif**, saves the control's content in **JPEG** format. • ***.gif**, , saves the control's content in **GIF** format. • ***.tif *.tiff**, saves the control's content in **TIFF** format. • ***.png**, saves the control's content in **PNG** format. • ***.pdf**, saves the control's content to PDF format. The File argument may carry up to 4 parameters separated by the | character in the following order: ***filename.pdf | paper size | margins | options***. In other words, you can specify the file name of the PDF document, the paper size, the margins and options to build the PDF document. By default, the paper size is 210 **mm** × 297 **mm** (A4 format) and the margins are 12.7 **mm** 12.7 **mm** 12.7 **mm** 12.7 **mm**. The units for the paper size and margins can be **pt** for PostScript Points, **mm** for Millimeters, **cm** for Centimeters, **in** for Inches and **px** for pixels. If PostScript Points are used if unit is missing. For instance, 8.27 in x 11.69 in, indicates the size of the paper in inches. Currently, the options can be **single**, which indicates that the control's content is exported to a single PDF page. For instance, the CopyTo("shot.pdf|33.11 in x 46.81 in|0 0 0 0|single") exports the control's content to an A0 single PDF page, with no margins. • ***.emf** or any other extension determines the control to

	save the control's content in **EMF** format. For instance, the CopyTo("c:\temp\snapshot.png") property saves the control's content in PNG format to snapshot.png file.
Variant	A boolean expression that indicates whether the File was successful saved, or a one dimension safe array of bytes, if the File parameter is empty string.

The CopyTo method copies/exports the control's view to BMP, PNG, JPG, GIF, TIFF, PDF or EMF graphical files, including no scroll bars. Use the Copy method to copy the control's content to the clipboard.

- The **BMP** file format, also known as bitmap image file or device independent bitmap (DIB) file format or simply a bitmap, is a raster graphics image file format used to store bitmap digital images, independently of the display device (such as a graphics adapter)
- The **JPEG** file format (seen most often with the .jpg extension) is a commonly used method of lossy compression for digital images, particularly for those images produced by digital photography.
- The **GIF** (Graphics Interchange Format) is a bitmap image format that was introduced by CompuServe in 1987 and has since come into widespread usage on the World Wide Web due to its wide support and portability.
- The **TIFF** (Tagged Image File Format) is a computer file format for storing raster graphics images, popular among graphic artists, the publishing industry, and both amateur and professional photographers in general.
- The **PNG** (Portable Network Graphics) is a raster graphics file format that supports lossless data compression. PNG was created as an improved, non-patented replacement for Graphics Interchange Format (GIF), and is the most used lossless image compression format on the Internet
- The **PDF** (Portable Document Format) is a file format used to present documents in a manner independent of application software, hardware, and operating systems. Each PDF file encapsulates a complete description of a fixed-layout flat document, including the text, fonts, graphics, and other information needed to display it.
- The EMF (Enhanced Metafile Format) is a 32-bit format that can contain both vector information and bitmap information. This format is an improvement over the Windows Metafile Format and contains extended features, such as the following

  Built-in scaling information
  Built-in descriptions that are saved with the file
  Improvements in color palettes and device independence

The EMF format is an extensible format, which means that a programmer can modify the original specification to add functionality or to meet specific needs. You can paste this format to Microsoft Word, Excel, Front Page, Microsoft Image Composer and any application that know to handle EMF formats.

The following VB sample saves the control's content to a file:

```
If (ExFileView1.CopyTo("c:\temp\test.emf")) Then
   MsgBox "test.emf file created, open it using the mspaint editor."
End If
```

The following VB sample prints the EMF content (as bytes, File parameter is empty string):

```
Dim i As Variant
For Each i In ExFileView1.CopyTo("")
   Debug.Print i
Next
```

method ExFileView.Debug ([dwReserved as Variant])

Displays information in debug mode.

Type	Description
dwReserved as Variant	Only for internal use.

The Debug property is not implemented in the registered version. The Debug method displays information only for debug configuration.

property ExFileView.DefaultItemHeight as Long

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

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

By default, the DefaultItemHeight property is 18 pixels. Use the DefaultItemHeight property to specify the height of the items. Use the Refresh method to update the control's content after changing the DefaultItemHeight property. The Font property specifies the control's font.

property ExFileView.Description(Type as DescriptionTypeEnum) as String

Changes descriptions for control objects.

Type	Description
Type as DescriptionTypeEnum	A long expression that defines the part being changed.
String	A string expression that indicates the part's description.

Use the Description property to customize the captions for control filter bar window. For instance, the Description(exFilterAll) = "(Include All)" changes the "(All)" item description in the filter bar window. Use the Description property to change the predefined strings in the filter bar window. Use the FilterBarCaption property to change the caption of the control's filter bar.

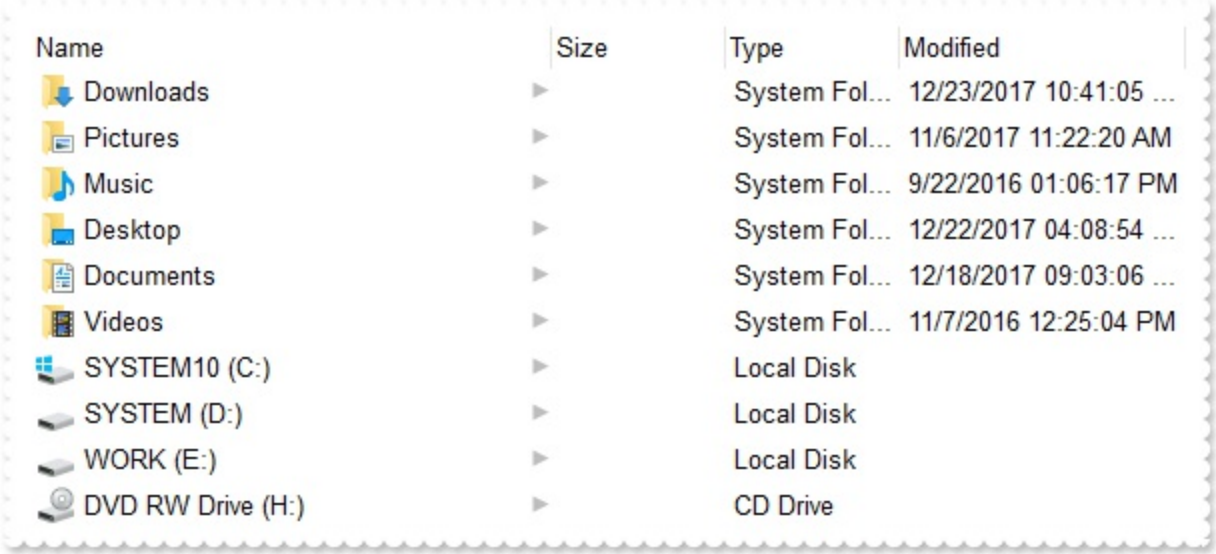
property ExFileView.DisplayFoldersInfo as Boolean

Specifies whether the control displays the Size, Type, Modified for folders objects.

Type	Description
Boolean	A Boolean expression that specifies whether the control displays the Size, Type, Modified for folders objects.

By default, the DisplayFoldersInfo property is True. If False, the control displays no information into the Size, Type, Modified columns for all folders objects as you can see in the following screen shot:

While True, the control should looks as:



property ExFileView.Enabled as Boolean

Enables or disables the control.

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

Use the Enabled property to disable the control. Use the ForeColor property to specify the control's foreground color. Use the BackColor property to specify the control's background color. Use the Font property to specify the control's font. Use the BackColorHeader property to specify the background color for the control's header bar. Use the ForeColorHeader properties to specify the foreground color for the control's header bar.

The following VB sample disables the control:

```
ExFileView1.Enabled = False
```

The following C++ sample disables the control:

```
m_fileview.SetEnabled( FALSE );
```

The following VB.NET sample disables the control:

```
AxExFileView1.Enabled = False
```

The following C# sample disables the control:

```
axExFileView1.Enabled = false;
```

The following VFP sample disables the control:

```
With thisform.ExFileView1
   .Object.Enabled = False
EndWith
```

method ExFileView.EndUpdate ()

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

Type	Description

property ExFileView.EventParam(Parameter as Long) as Variant

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

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

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

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

In most languages you will type something like:

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

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

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

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

reference.

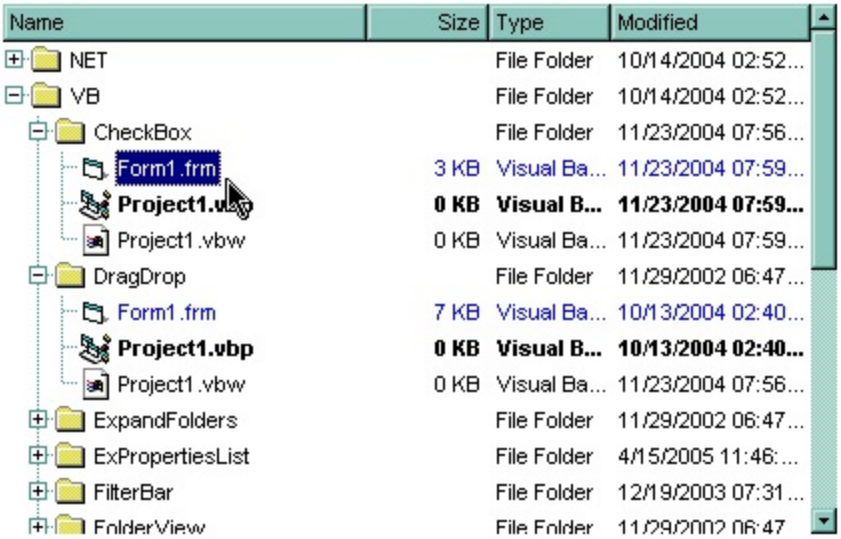
Calling the EventParam property outside of an event produces an OLE error, such as pointer invalid, as its scope was designed to be used only during events.

property ExFileView.ExcludeFilter as String

Specifies the pattern used to exclude files from the control's list, like '*.tmp *.log'.

Type	Description
String	A string expression that may contains wild cards like * or ?

Use the ExcludeFilter property to exclude files that match a pattern or multiple patterns from the current list. When the ExcludeFilter is set, the control automatically refreshes the current list, and applies the FileType attributes. Use the IncludeFilter property to include only the files that match a pattern. To remove a previous exclude filter you can use ("") empty string. By default, the ExcludeFilter property is "". The IncludeFolderFilter property specifies a wild characters expression that indicates the folders being included. The ExcludeFolderFilter property specifies a wild characters expression that indicates the folders being excluded. The patterns are separated by space character. Use the exHideFileExtensionsForKnownFileTypes option to show the file extensions in case your Windows Explorer, the "**Hide File Extensions For Known File Types**" is checked. Use the Name property to specify the name of the file or the folder. Use the Folder property to specify whether the File object holds a file or a folder.



The following VB sample excludes the files with the extensions: 'bat', 'com' and 'sys':

```
With ExFileView1
   .ExcludeFilter = "*.bat *.sys *.com"
End With
```

The following C++ sample excludes the files with the extensions: 'bat', 'com' and 'sys':

```
m_fileview.SetExcludeFilter( "*.bat *.sys *.com" );
```

The following VB.NET sample excludes the files with the extensions: 'bat', 'com' and 'sys':

```
AxExFileView1.ExcludeFilter = "*.bat *.sys *.com"
```

The following C# sample excludes the files with the extensions: 'bat', 'com' and 'sys':

```
axExFileView1.ExcludeFilter = "*.bat *.sys *.com";
```

The following VFP sample excludes the files with the extensions: 'bat', 'com' and 'sys':

```
With thisform.ExFileView1
   .ExcludeFilter = "*.bat *.sys *.com"
EndWith
```

property ExFileView.ExcludeFolderFilter as String

Retrieves or sets a value that indicates the folders being excluded.

Type	Description
String	A string expression that may contains wild cards like * or ?

Use the ExcludeFolderFilter property to exclude folders that match a pattern or multiple patterns from the current list. The ExcludeFolderFilter property has ni effect if it is empty. When the ExcludeFolderFilter is set, the control automatically refreshes the current list, and applies the FileType attributes. Use the ExcludeFilter property to exclude files that match a pattern or multiple patterns from the current list. Use the IncludeFilter property to include only the files that match a pattern. Use the exHideFileExtensionsForKnownFileTypes option to show the file extensions in case your Windows Explorer, the "**Hide File Extensions For Known File Types**" is checked. Use the Name property to specify the name of the file or the folder. Use the Folder property to specify whether the File object holds a file or a folder.

The following VB sample excludes the "Temp" folders:

```
With ExFileView1
   .ExcludeFolderFilter = "*temp*"
End With
```

The following C++ sample excludes the "Temp" folders:

```
m_fileview.SetExcludeFolderFilter( "temp*" );
```

The following VB.NET sample excludes the "Temp" folders:

```
AxExFileView1.ExcludeFolderFilter = "*temp*"
```

The following C# sample excludes the "Temp" folders:

```
axExFileView1.ExcludeFolderFilter = "*temp*";
```

The following VFP sample excludes the "Temp" folders:

```
With thisform.ExFileView1
   .ExcludeFolderFilter = "*temp*"
EndWith
```

method ExFileView.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. Use the Get property to retrieve the selected item(s). Use the Name property to specify the name of the file or the folder. Use the Folder property to specify whether the File object refers a file or a folder.

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)**

The following VB sample opens that file being double clicked:

```
Private Sub ExFileView1_DblClick()
  With ExFileView1.Get(SelItems)
    If (.Count > 0) Then
      With .Item(0)
        If (Not .Folder) Then
          ExFileView1.ExecuteContextCommand .Name, .Folder, "Open"
        End If
      End With
    End If
  End With
```

```
End Sub
```

The following VB sample opens a file that user double clicks (the sample uses the Open's identifier, in case your application is not displaying the English messages) :

```
Private Sub ExFileView1_DblClick()
   With ExFileView1.Get(SelItems)
      If (.Count > 0) Then
         With .Item(0)
            If (Not .Folder) Then
               ExFileView1.ExecuteContextCommand .Name, .Folder, "102"
            End If
         End With
      End If
   End With
End Sub
```

The following C++ sample opens that file being double clicked:

```
void OnDblClickExfileview1()
{
   CFiles files = m_fileview.GetGet( 0 );
   if ( files.GetCount() > 0 )
   {
      CFile1 file = files.GetItem( COleVariant( (long)0 ) );
      m_fileview.ExecuteContextCommand( file.GetName(), file.GetFolder(), "Open" );
   }
}
```

The following VB.NET sample opens that file being double clicked:

```
Private Sub AxExFileView1_DblClick(ByVal sender As Object, ByVal e As System.EventArgs)
Handles AxExFileView1.DblClick
   With AxExFileView1.get_Get(EXFILEVIEWLib.TypeEnum.SelItems)
      If (.Count > 0) Then
         With .Item(0)
            If (Not .Folder) Then
               AxExFileView1.ExecuteContextCommand(.Name, .Folder, "Open")
            End If
```

```
        End With
      End If
    End With
End Sub
```

The following C# sample opens that file being double clicked:

```
private void axExFileView1_DblClick(object sender, EventArgs e)
{
    EXFILEVIEWLib.Files files = axExFileView1.get_Get(EXFILEVIEWLib.TypeEnum.SelItems);
    if (files.Count > 0)
    {
        EXFILEVIEWLib.File file = files[0];
        axExFileView1.ExecuteContextCommand(file.Name, file.Folder, "Open");
    }
}
```

The following VFP sample opens that file being double clicked:

```
*** ActiveX Control Event ***

With thisform.ExFileView1.Get(0)   && SelItems
    If (.Count > 0) Then
        With .Item(0)
            If (Not .Folder) Then
                thisform.ExFileView1.ExecuteContextCommand(.Name, .Folder, "Open")
            EndIf
        EndWith
    EndIf
EndWith
```

property ExFileView.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 AllowMenuContext property specifies whether the control shows the object's context menu when the user presses the right click over a file or folder.

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

```
Private Sub ExFileView1_StateChange(ByVal State As EXFILEVIEWLibCtl.StateChangeEnum)
   With ExFileView1
      If (State = ShowContextMenu) Then
         .ShowContextMenu = .ShowContextMenu + ",Item 1[id=1][def],Popup[id=2](Sub-
Item 2[id=2],[sep],Sub-Item 3[id=3])"
      Else
         If (State = ExecuteContextMenu) Then
            Debug.Print "You selected the command: " & .ExecuteContextMenu
         End If
      End If
   End With
End Sub
```

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

```
Private Sub ExFileView1_StateChange(ByVal State As EXFILEVIEWLibCtl.StateChangeEnum)
   With ExFileView1
      If (State = ExecuteContextMenu) Then
         If Not (.ExecuteContextMenu = 17) Then   ' Delete
            Debug.Print "You selected the command: " & .ExecuteContextMenu
         Else
            .ExecuteContextMenu = 0
```

```
            MsgBox "Delete is disabled."
        End If
    End If
  End With
End Sub
```

method ExFileView.ExecuteTemplate (Template as String)

Executes a template and returns the result.

Type	Description
Template as String	A Template string being executed

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

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

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

```
Debug.Print ExFileView1.ExecuteTemplate("Items.ItemBar(FirstVisibleItem(),``,1)")
```

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

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

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

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

"\n\r" (newline) characters.

An instruction can be one of the following:

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

The Template supports the following general functions:

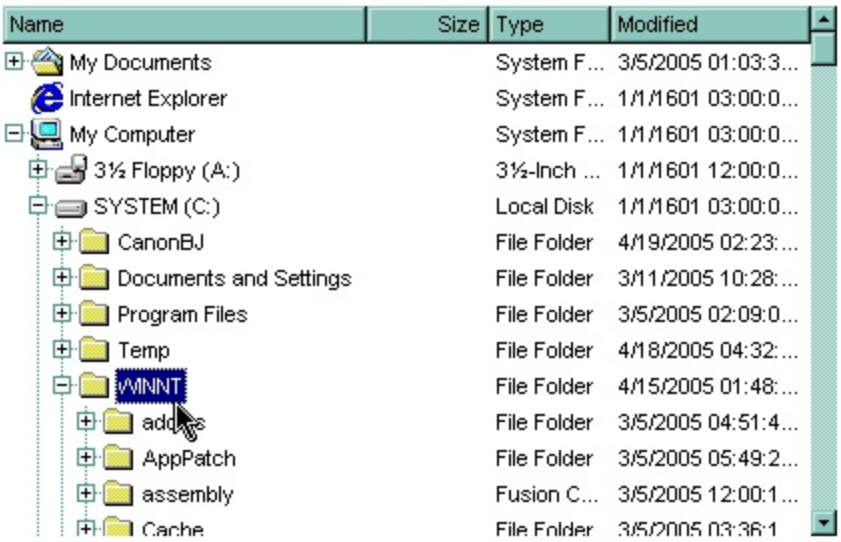
- **RGB(**R,G,B**)** *property retrieves an RGB value, where the R, G, B are byte values that indicates the R G B values for the color being specified. For instance, the following code changes the control's background color to red: BackColor = RGB(255,0,0)*
- **CreateObject(**progID**)** *property creates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

method ExFileView.Expand (Folder as String)

Expands and selects a folder giving its path.

Type	Description
Folder as String	A string expression that indicates the name of the folder being expanded, the relative path of the folder being expanded, or the absolute path of the folder being expanded. If the Folder parameter is "*", all visible folders are recursively expanded, equivalent of expand all.

Use the Expand method to expand and select a folder giving its path. The absolute path starts with the letter drive (like c:\ ). Any path is separated by the / character. The Expand method retrieves no error if it is not able to find the folder. If a relative or absolute path is giving, the control expands all found folders. Use the ExpandFolders property to assign a + sign to folders that includes sub folders. Use the ExpandOnDblClk property to expand or collapse a folder when user double clicks the folder. Use the ExploreFromHere property specifies the root folder for the control. Use the Get method to retrieve the collection of selected items.



The following VB sample expands the "winnt\system32" (relative path) folder:

```
With ExFileView1
   .ExpandFolders = True
   .HasLinesAtRoot = True
   .Expand "WINNT\system32"
End With
```

The following VB sample expands the "c:\winnt" (absolute path) folder when the control is browsing the 'Desktop':

```
With ExFileView1
   .ExpandFolders = True
   .ExploreFromHere = "::{00021400-0000-0000-C000-000000000046}"
   .Expand "C:\WINNT"
End With
```

The following C++ sample expands the "c:\winnt" folder:

```
m_fileview.SetExpandFolders( TRUE );
m_fileview.SetExploreFromHere( "::{00021400-0000-0000-C000-000000000046}" );
m_fileview.Expand( "c:\\winnt" );
```

The following VB.NET sample expands the "c:\winnt" folder:

```
With AxExFileView1
   .ExpandFolders = True
   .ExploreFromHere = "::{00021400-0000-0000-C000-000000000046}"
   .Expand("C:\WINNT")
End With
```

The following C# sample expands the "c:\winnt" folder:

```
axExFileView1.ExpandFolders = true;
axExFileView1.ExploreFromHere = "::{00021400-0000-0000-C000-000000000046}";
axExFileView1.Expand("C:\\WINNT");
```

The following VFP sample expands the "c:\winnt" folder:

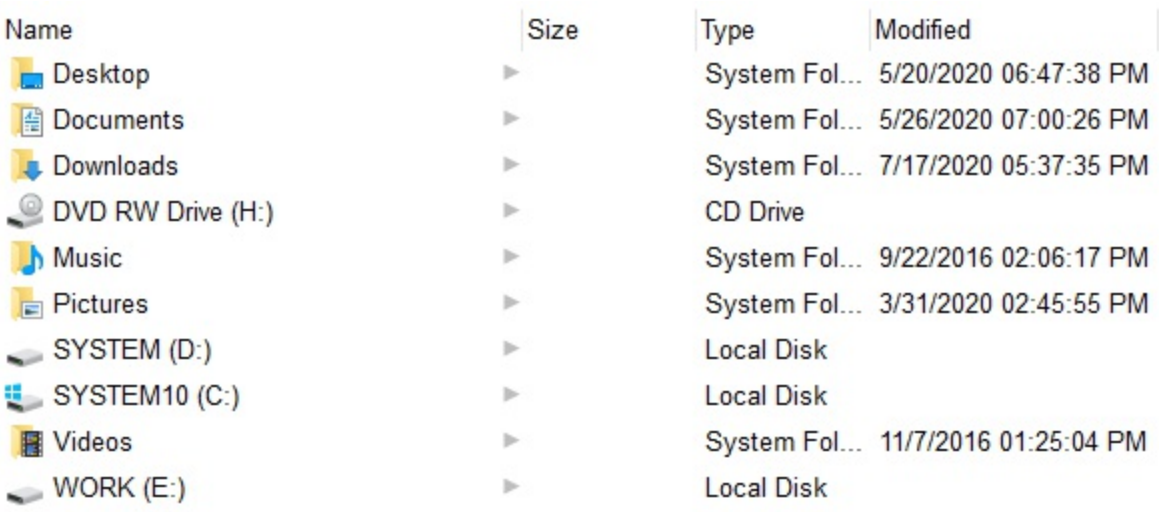
```
With thisform.ExFileView1
   .ExpandFolders = .t.
   .ExploreFromHere = "::{00021400-0000-0000-C000-000000000046}"
   .Expand("C:\WINNT")
EndWith
```

property ExFileView.ExpandFolders as Boolean

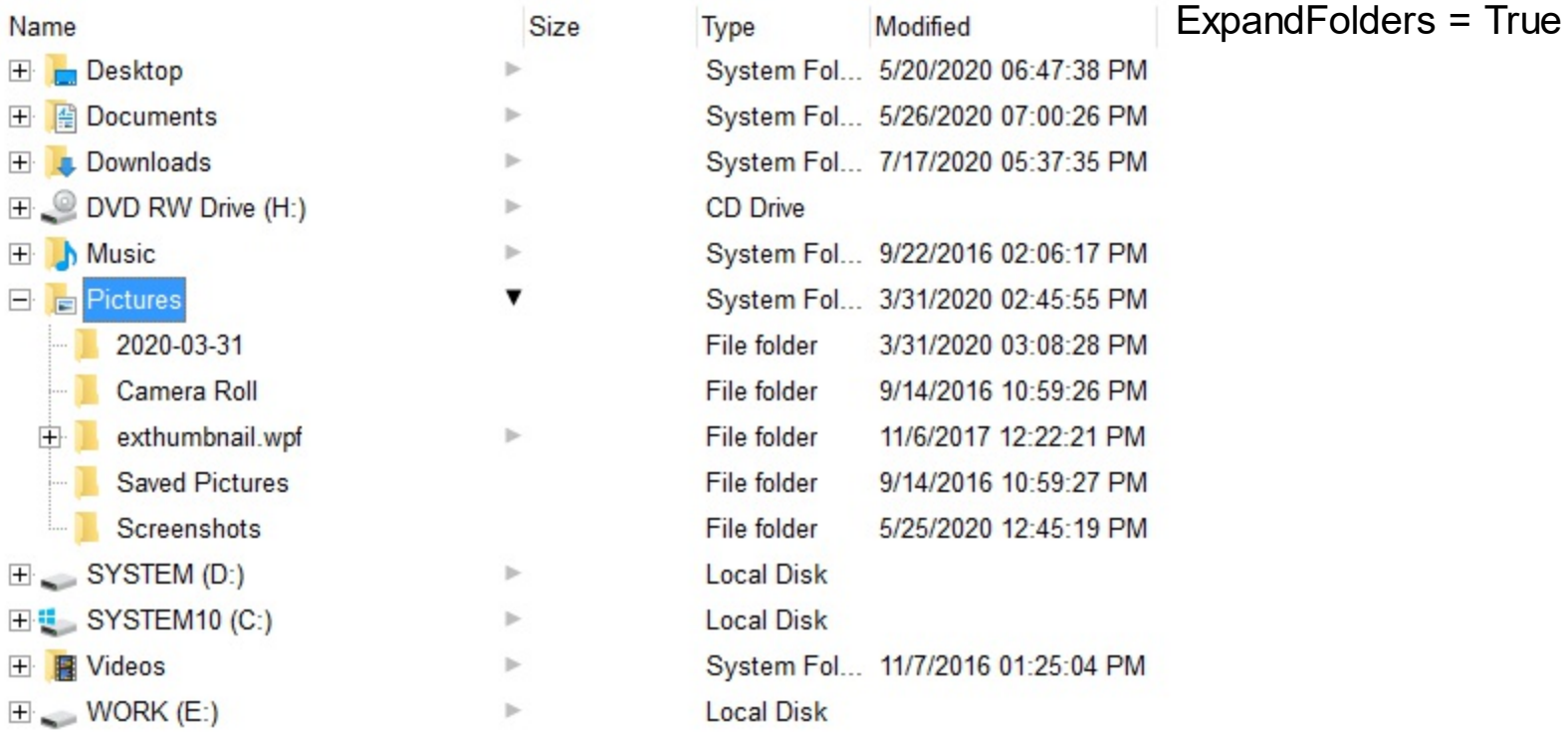
Retrieves or sets a value that indicates whether the control expands the folder objects.

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

By default, the ExpandFolders property is False. If the ExpandFolders property is True, each folder that contains a subfolder, displays +/- button, that allows to expand or collapse the folder. You can use the ExpandFolder property to let ExFileView control simulates a folderview control. Use the IncludeFolders property to include folders in the current list. Use the HasButtons property to hide or show the + buttons. Use the IncludeFilesInFolder property to include files when expanding a folder. Use the Expand method to programmatically expand a folder giving its path. Use the ExpandOnDblClk property to expand or collapse a folder when user double clicks the folder. Use the ExploreFromHere property specifies the root folder for the control. The Folder property specifies whether a File object holds a file or a folder. The IncludeFolderFilter property specifies a wild characters expression that indicates the folders being included. The ExcludeFolderFilter property specifies a wild characters expression that indicates the folders being excluded. The Indent property retrieves or sets the amount, in pixels, that child items are indented relative to their parent items.



ExpandFolders = False



The File.Children property helps you to collect recursively all files/folders of specified object. The EnumR function displays the full name of each file/folder, and goes recursively to each subfolder. Use the Folder property to specify whether the File object hosts a file or a folder. The Children property returns a collection of File objects, if the ExpandFolders property is True.

```
Public Sub EnumR(ByVal f As EXFILEVIEWLibCtl.File)
   Debug.Print f.FullName
   For Each c In f.Children
      EnumR (c)
   Next
End Sub
```

The following VB sample expands the "c:\winnt" (absolute path) folder when the control is browsing the 'Desktop':

```
With ExFileView1
   .ExpandFolders = True
   .ExploreFromHere = "::{00021400-0000-0000-C000-000000000046}"
   .Expand "C:\WINNT"
End With
```

The following C++ sample expands the "c:\winnt" folder:

```
m_fileview.SetExpandFolders( TRUE );
m_fileview.SetExploreFromHere( "::{00021400-0000-0000-C000-000000000046}" );
m_fileview.Expand( "c:\\winnt" );
```

The following VB.NET sample expands the "c:\winnt" folder:

```
With AxExFileView1
   .ExpandFolders = True
   .ExploreFromHere = "::{00021400-0000-0000-C000-000000000046}"
   .Expand("C:\WINNT")
End With
```

The following C# sample expands the "c:\winnt" folder:

```
axExFileView1.ExpandFolders = true;
axExFileView1.ExploreFromHere = "::{00021400-0000-0000-C000-000000000046}";
axExFileView1.Expand("C:\\WINNT");
```

The following VFP sample expands the "c:\winnt" folder:

```
With thisform.ExFileView1
   .ExpandFolders = .t.
   .ExploreFromHere = "::{00021400-0000-0000-C000-000000000046}"
   .Expand("C:\WINNT")
EndWith
```

property ExFileView.ExpandOnDblClk as Boolean

Retrieves or sets a value that indicates whether a folder is expanded by double click.

Type	Description
Boolean	A boolean expression that indicates whether a folder is expanded by double click.

By default, the ExpandOnDblClick property is False. Use the ExpandOnDblClk property to expand or collapse a folder when user double clicks a folder. When ExpandOnDblClick property is False, the control browses for a new folder when user double clicks the folder. Use the Expand method to programmatically expand a folder giving its path. Use the ExpandFolders property to assign a + sight to folders that include sub folders. The control fires the DblClick event when the user double clicks a file or a folder. Use the FileFromPoint property to retrieve the file from the cursor. Use the Get property to retrieve the selected item.

property ExFileView.ExploreFromHere as String

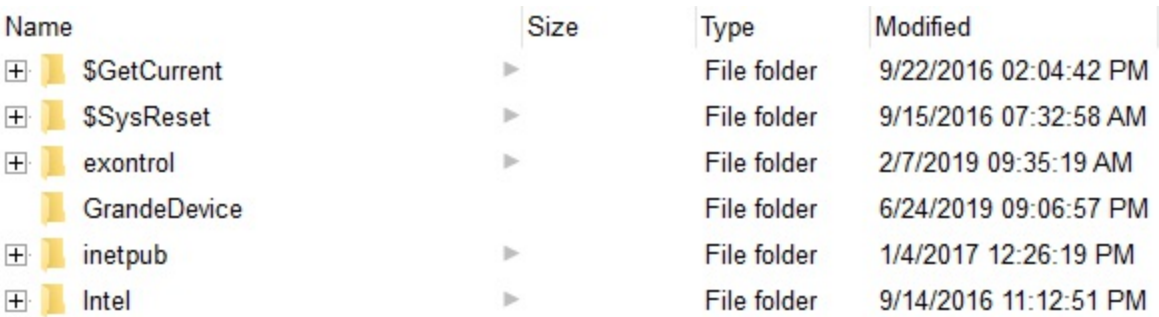
Specifies the root folder(s) for the control.

Type	Description
String	A string expression that indicates the folder's path that's the root of the control.

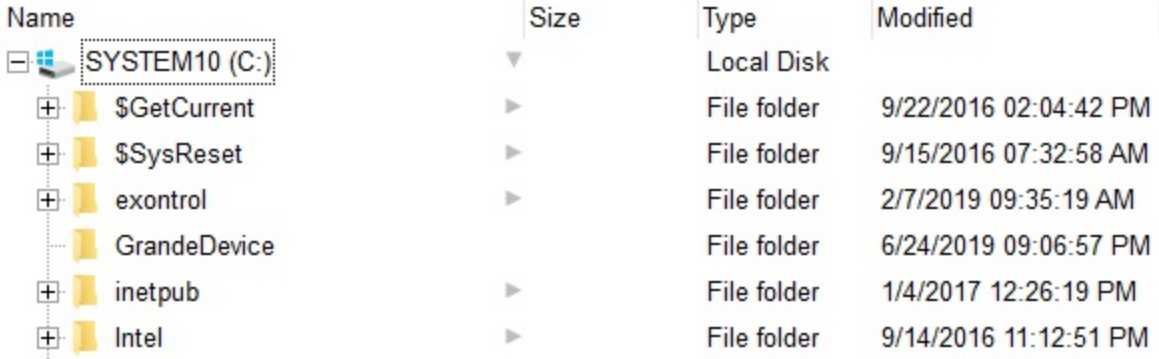
By default, the ExploreFromHere property is "C:\". The ExplorerFromHere property specifies the root folder(s) for the control. The ExpandFolders property retrieves or sets a value that indicates whether the control expands the folder objects. Use the IncludeFolders property to exclude folders from the current list.

Starting with the version **14.0**, the control supports multiple root-folders. The ExploreFromHere property specifies multiple root-folders if they are separated by | or \r\n characters. If the ExploreFromHere entity includes a > character, the characters after indicates the HTML caption to be displayed instead of its default name. For instance: "C:\>" includes the C system driver as a root folder, rather than listing its content, "C:\>System C", includes the system C drive as a root folder with the name System C, c in bold, "C:\|E:\" specifies that the control includes C and E system drives, as root folders, "E:\Exontrol|::{20D04FE0-3AEA-1069-A2D8-08002B30309D}" specifies to include the E:\Exontrol folder and the My Computer system folder.

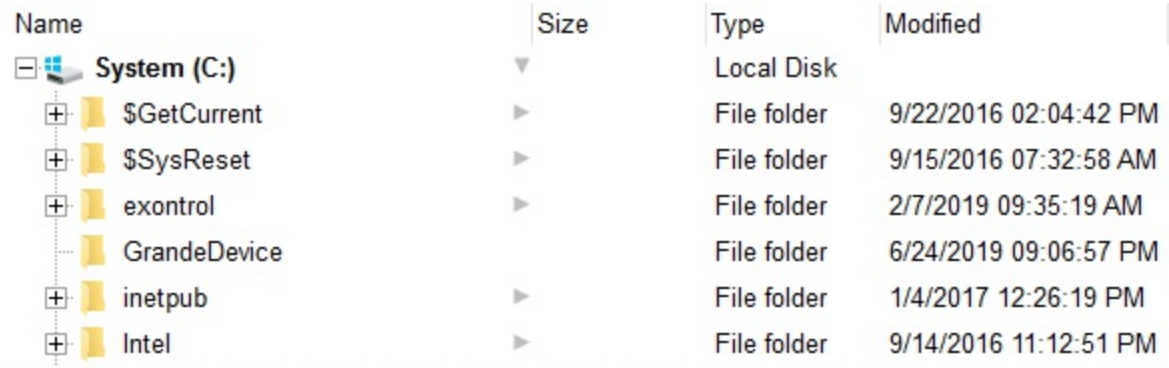
The following screen shots show the control ExploreFromHere property takes different values such as "C:\", "C:\>", "C:\>System (C:)" or "C:\|D:\|E:\Exontrol"



ExploreFromHere = "C:\"

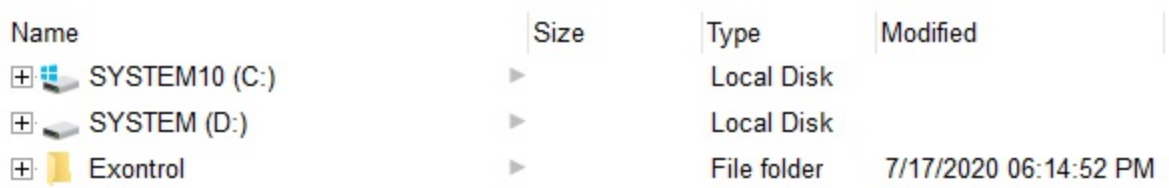


ExploreFromHere = "C:\>"



ExploreFromHere =

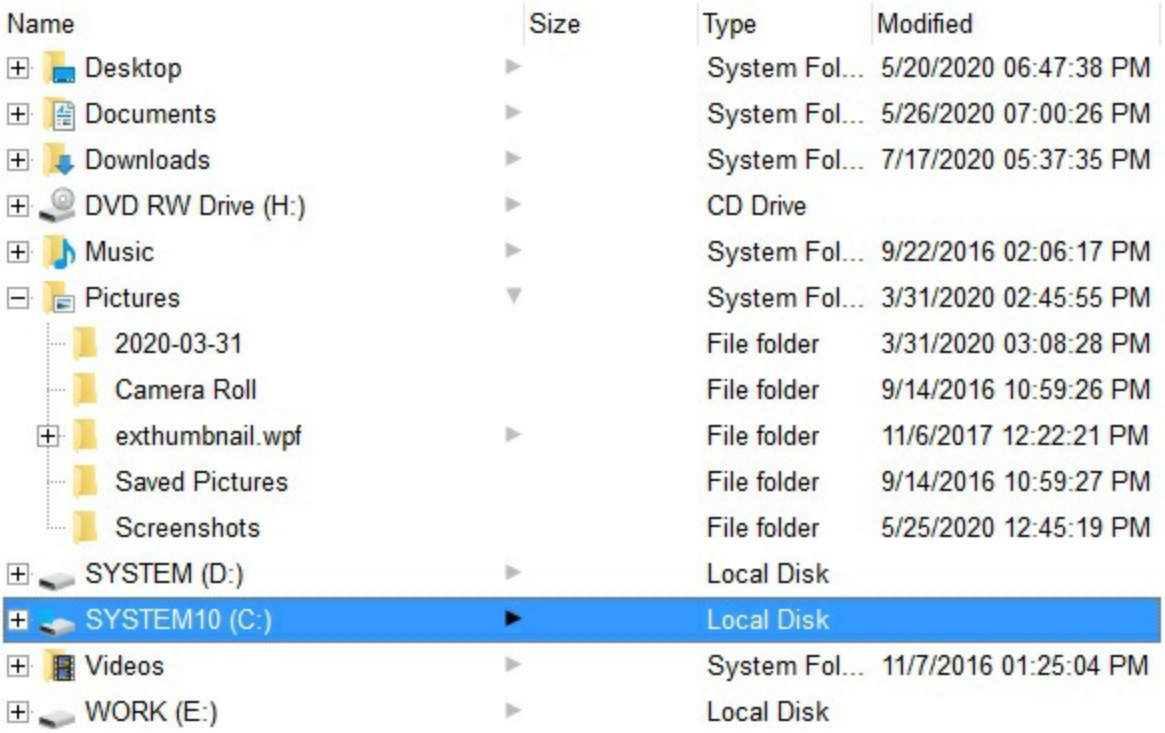
"C:\>System (C:)"



ExploreFromHere =

"C:\|D:\|E:\Exontrol"

The ExploreFromHere property changes the BrowseFolderPath property. The BrowseFolderPath property retrieves or sets the browsed folder path. *The BrowseFolderPath property has no effect (returns empty string), if the control's ExploreFromHere property includes |, > or \r\n characters (shortly the BrowseFolderPath property has no effect if the control display multiple root-folders).* Use the HasButtons property to hide or show the + buttons. Use the IncludeFilesInFolder property to include files when expanding a folder. Use the Expand method to programmatically expand a folder giving its path. Use the Add method to add rules to highlight the files and folders in the control. The RelativeName property gets the relative path for the file or folder, based on the ExploreFromHere property.



The following VB samples browses the "c:\temp" folder:

```
ExFileView1.ExploreFromHere = "c:\temp"
```

The following VB samples browses the "Desktop" folder:

```
ExFileView1.ExploreFromHere = "::{00021400-0000-0000-C000-000000000046}"
```

The following VB samples browses the "My Network Places" folder:

```
ExFileView1.ExploreFromHere = "::{208D2C60-3AEA-1069-A2D7-08002B30309D}"
```

The following VB samples browses the "My Computer" folder:

```
ExFileView1.ExploreFromHere = "::{20D04FE0-3AEA-1069-A2D8-08002B30309D}"
```

The following C++ samples browses the "Desktop" folder:

```
m_fileview.SetExploreFromHere( "::{00021400-0000-0000-C000-000000000046}" );
```

The following VB.NET samples browses the "Desktop" folder:

```
With AxExFileView1
   .ExploreFromHere = "::{00021400-0000-0000-C000-000000000046}"
End With
```

The following C# samples browses the "Desktop" folder:

```
axExFileView1.ExploreFromHere = "::{00021400-0000-0000-C000-000000000046}";
```

The following VFP samples browses the "Desktop" folder:

```
With thisform.ExFileView1
   .ExploreFromHere = "::{00021400-0000-0000-C000-000000000046}"
EndWith
```

property ExFileView.FileFromPoint (X as OLE_XPOS_PIXELS, Y as OLE_YPOS_PIXELS) as String

Retrieves the file from the 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 indicates the path of the file from the cursor.

Use the FileFromPoint property to get the path of the file from the cursor. The control fires the StateChange event when the user selects a file or a folder. Use Get property to retrieve the collection of selected items. Use the BrowseFolderPath property to retrieve the browsed folder path.

The following VB sample displays the file from the cursor:

```
Private Sub ExFileView1_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
   Dim f As String
   f = ExFileView1.FileFromPoint(X / Screen.TwipsPerPixelX, Y / Screen.TwipsPerPixelY)
   If Len(f) > 0 Then
      Debug.Print f
   End If
End Sub
```

The following C++ sample displays the file from the cursor:

```
void OnMouseMoveExfileview1(short Button, short Shift, long X, long Y)
{
   CString f = m_fileview.GetFileFromPoint( X, Y );
   if ( f.GetLength() > 0 )
      OutputDebugString( f );
}
```

The following VB.NET sample displays the file from the cursor:

```
Private Sub AxExFileView1_MouseMoveEvent(ByVal sender As Object, ByVal e As
AxEXFILEVIEWLib._IExFileViewEvents_MouseMoveEvent) Handles
AxExFileView1.MouseMoveEvent
   Dim f As String = AxExFileView1.get_FileFromPoint(e.x, e.y)
   If Len(f) > 0 Then
      Debug.WriteLine(f)
   End If
End Sub
```

The following C# sample displays the file from the cursor:

```
private void axExFileView1_MouseMoveEvent(object sender,
AxEXFILEVIEWLib._IExFileViewEvents_MouseMoveEvent e)
{
   string f = axExFileView1.get_FileFromPoint(e.x, e.y);
   if (f.Length > 0)
      System.Diagnostics.Debug.WriteLine(f);
}
```

The following VFP sample displays the file from the cursor:

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

with thisform.ExFileView1
   local f
   f = .FileFromPoint( x, y )
   if ( len(f) > 0 )
      wait window nowait f
   endif
endwith
```

property ExFileView.FileTypes as FileTypes

Retrieves the control's FileTypes collection.

Type	Description
FileTypes	A FileTypes collection associated to the control.

Use the FileTypes property to access the control's FileType objects. Use the Add method to add new rules for the control's content. Use the Apply method to apply the rules. Use the Font property to specify the control's font. Use the ForeColor property to specify the control's foreground color. Use the BackColor property to specify the control's background color. Use the Get property to get the list of files and folders.

The following VB sample bolds the cpp and h files:

```
With ExFileView1.FileTypes.Add("*.cpp *.h")
   .Bold = True
   .Apply
End With
```

The following C++ sample bolds the cpp and h files:

```
#include "FileType.h"
#include "FileTypes.h"
CFileType fileType = m_fileview.GetFileTypes().Add("*.cpp *.h");
fileType.SetBold( TRUE );
fileType.Apply();
```

The following VB.NET sample bolds the cpp and h files:

```
With AxExFileView1.FileTypes.Add("*.cpp *.h")
   .Bold = True
   .Apply()
End With
```

The following C# sample bolds the cpp and h files:

```
EXFILEVIEWLib.FileType fileType = axExFileView1.FileTypes.Add("*.cpp *.h");
fileType.Bold = true;
fileType.Apply();
```

The following VFP sample bolds the cpp and h files:

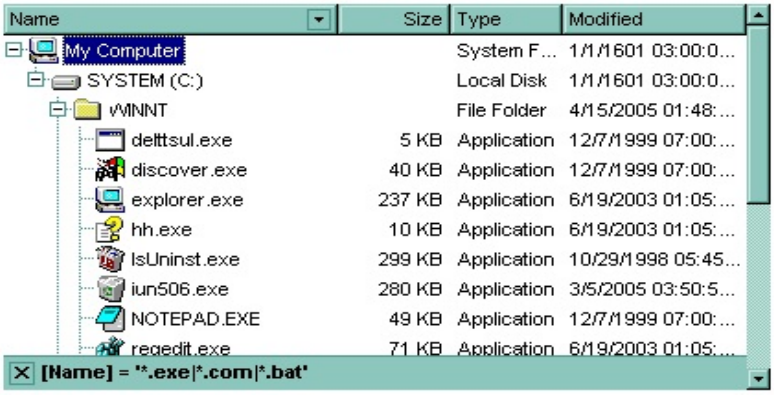
```
With thisform.ExFileView1.FileTypes.Add("*.cpp *.h")
   .Bold = .t.
   .Apply()
EndWith
```

property ExFileView.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 FilterBarHeight property to hide the control's filter bar header. Use the BackColor property to specify the control's background color.

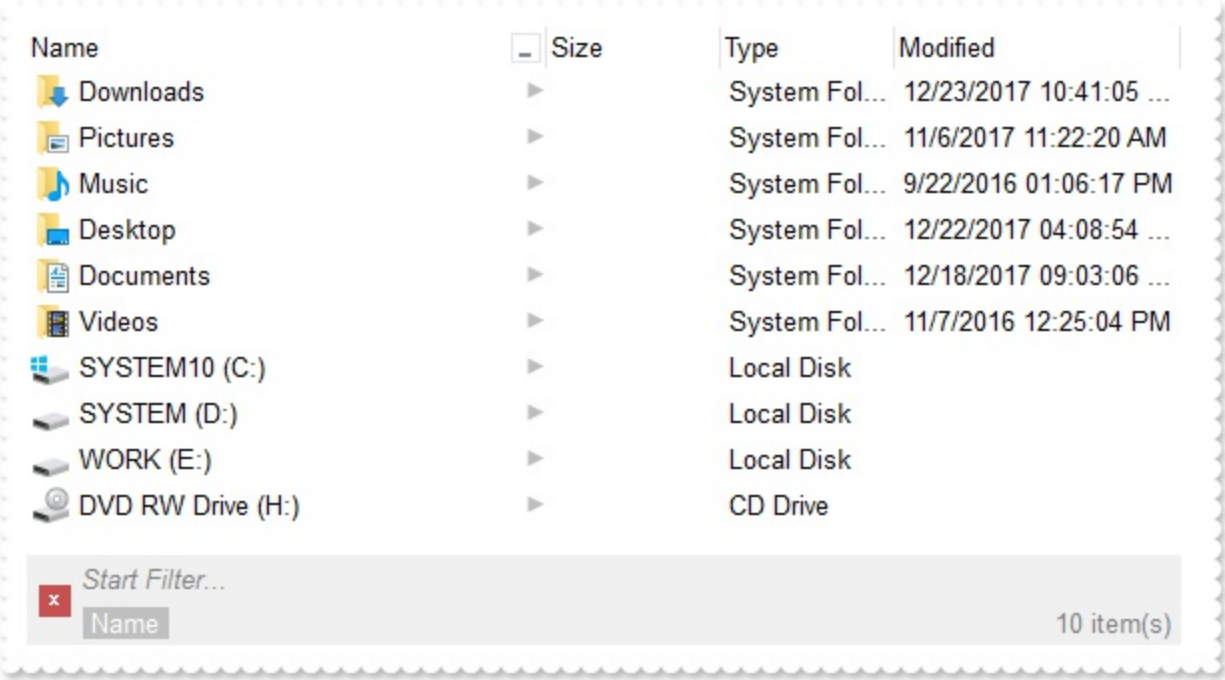


property ExFileView.FilterBarCaption as String

Specifies the filter bar's caption.

Type	Description
String	A string value that defines the expression to display the control's filter bar.

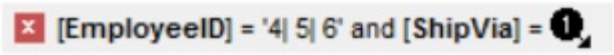
By default, the FilterBarCaption property is empty. You can use the FilterBarCaption property to define the way the filter bar's caption is being displayed. The FilterBarCaption is displayed on the bottom side of the control where the control's filter bar is shown. While the FilterBarCaption property is empty, the control automatically builds the caption to be displayed on the filter bar from all columns that participates in the filter using its name and values. The FilterBarCaption property supports expressions as explained bellow.



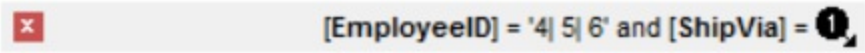
For instance:

- "no filter", shows no filter caption all the time

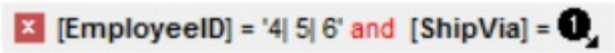
- "" displays no filter bar, if no filter is applied, else it displays the current filter



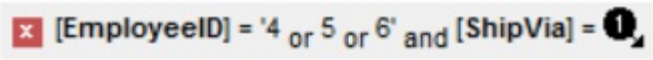
- "`<r>` + value", displays the current filter caption aligned to the right. You can include the exFilterBarShowCloseOnRight flag into the FilterBarPromptVisible property to display the close button aligned to the right



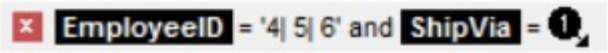
- "value replace ` and ` with `<fgcolor=FF0000> and </fgcolor>`", replace the AND keyword with a different foreground color



- "value replace ` and ` with `<off 4> and </off>` replace `|` with ` <off 4>or</off> ` replace ` ` with ` `", replaces the AND and | values

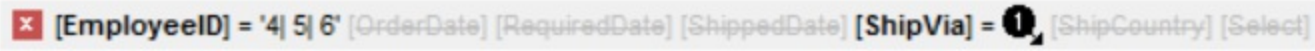


- "value replace `[` with `<bgcolor=000000><fgcolor=FFFFFF><b> ` replace `]` with ` </b></bgcolor></fgcolor>`", highlights the columns being filtered with a different background/foreground colors.



- "value + ` ` + available", displays the current filter, including all available columns to be filtered

- "allui" displays all available columns



- "((allui + `<fgcolor=808080>` + (matchitemcount < 0 ? ((len(allui) ? `` : ``) + `<r>` + abs(matchitemcount + 1) + ` result(s)`) : (`<r><fgcolor=808080>`+ itemcount + ` item(s)`))) replace `[<b>` with `<bgcolor=000000><fgcolor=FFFFFF><b> ` replace `</b>]` with ` </b></bgcolor></fgcolor>` replace `[<s>` with `<bgcolor=C0C0C0> <fgcolor=FFFFFF> ` replace `</s>]` with ` </bgcolor></fgcolor>`)" displays all available columns to be filtered with different background/foreground colors including the number of items/results

Use the FilterBarForeColor and FilterBarBackColor properties to define the colors used to paint the description for control's filter. Use the FilterBarHeight property to specify the height of the control's filter bar. Use the FilterBarFont property to specify the font for the control's filter bar. Use the Description property to define predefined strings in the filter bar

caption. The FilterBarPromptVisible property specifies whether how/where the control's filter/prompt is shown.

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

- **value** or **current** keyword returns the current filter as a string. At runtime the value may return a string such as "[EmployeeID] = '4| 5| 6' and [ShipVia] = <img>1</img>", so the control automatically applies HTML format, which you can change it. For instance, "upper(value)" displays the caption in uppercase or "value replace `<b>` with `<fgcolor=808080>` replace `</b>` with `</fgcolor>`" displays the column's name with a different foreground color.
- **itemcount** keyword returns the total number of items. At runtime the itemcount is a positive integer that indicates the count of all items. For instance, "value + `<r> <fgcolor=808080>Total: ` + itemcount" includes in the filter bar the number of items aligned to the right.
- **visibleitemcount** keyword returns the number of visible items. At runtime, the visibleitemcount is a positive integer if no filter is applied, and negative if a filter is applied. If positive, it indicates the number of visible items. The visible items does not include child items of a collapsed item. If negative, a filter is applied, and the absolute value minus one, indicates the number of visible items after filter is applied. 0 indicates no visible items, while -1 indicates that a filter is applied, but no item matches the filter criteria. For instance, "value + `<r><fgcolor=808080>` + (visibleitemcount < 0 ? (`Result: ` + (abs(visibleitemcount) - 1)) : (`Visible: ` + visibleitemcount))" includes "Visible: " plus number of visible items, if no filter is applied or "Result: " plus number of visible items, if filter is applied, aligned to the right
- **matchitemcount** keyword returns the number of items that match the filter. At runtime, the matchitemcount is a positive integer if no filter is applied, and negative if a filter is applied. If positive, it indicates the number of items within the control. If negative, a filter is applied, and the absolute value minus one, indicates the number of matching items after filter is applied. A matching item includes its parent items, if the control's FilterInclude property allows including child items. 0 indicates no visible items, while -1 indicates that a filter is applied, but no item matches the filter criteria. For instance, "value + `<r><fgcolor=808080>` + (matchitemcount < 0 ? (`Result: ` + (abs(matchitemcount) - 1)) : (`Visible: ` + matchitemcount))" includes "Visible: " plus number of visible items, if no filter is applied or "Result: " plus number of macthing items, if filter is applied, aligned to the right
- **leafitemcount** keyword returns the number of leaf items. A leaf item is an item with no child items. At runtime, the leafitemcount is a positive number that computes the number of leaf items (expanded or collapsed). For instance, the "value + `<r> <fgcolor=808080><font ;6>` + leafitemcount" displays the number of leaf items aligned to the right with a different font and foreground color.
- **promptpattern** returns the pattern in the filter bar's prompt, as a string. The

FilterBarPromptPattern specifies the pattern for the filter prompt. The control's filter bar prompt is visible, if the exFilterBarPromptVisible flag is included in the FilterBarPromptVisible property.

- **available** keyword returns the list of columns that are not currently part of the control's filter, but are available to be filtered. A column is available to be filtered, if the ColumnFilterButton property of the Column object, is True. At runtime, the available keyword may return a string such as "<fgcolor=C0C0C0>[<s>OrderDate</s>] <fgcolor> </fgcolor>[<s>RequiredDate</s>]<fgcolor> </fgcolor> [<s>ShippedDate</s>]<fgcolor> </fgcolor>[<s>ShipCountry</s>]<fgcolor> </fgcolor> [<s>Select</s>]</fgcolor>", so the control automatically applies HTML format, which you can change it. For instance, "value + ` ` + available", displays the current filter, including all available columns to be filtered. For instance, the "value + `<r>` + available replace `C0C0C0` with `FF0000`" displays the available columns aligned to the right with a different foreground color.
- **allui** keyword returns the list of columns that are part of the current filter and available columns to be filtered. A column is available to be filtered, if the ColumnFilterButton property of the Column object, is True. At runtime, the allui keyword may return a string such as "[EmployeeID] = '4| 5| 6'<fgcolor> </fgcolor><fgcolor=C0C0C0> [<s>OrderDate</s>]</fgcolor><fgcolor> </fgcolor><fgcolor=C0C0C0> [<s>RequiredDate</s>]</fgcolor><fgcolor> </fgcolor><fgcolor=C0C0C0> [<s>ShippedDate</s>]</fgcolor><fgcolor> </fgcolor>[ShipVia] = <img>1</img><fgcolor> </fgcolor><fgcolor=C0C0C0>[<s>ShipCountry</s>]</fgcolor> <fgcolor> </fgcolor><fgcolor=C0C0C0>[<s>Select</s>]</fgcolor>", so the control automatically applies HTML format, which you can change it. For instance, "allui", displays the current filter, including all available columns to be filtered. For instance, the "((allui + `<fgcolor=808080>` + (matchitemcount < 0 ? ((len(allui) ? `` : ``) + `<r>` + abs(matchitemcount + 1) + ` result(s)`) : (`<r><fgcolor=808080>`+ itemcount + ` item(s)`))) replace `[<b>` with `<bgcolor=000000><fgcolor=FFFFFF><b> ` replace `</b>]` with ` </b></bgcolor></fgcolor>` replace `[<s>` with `<bgcolor=C0C0C0> <fgcolor=FFFFFF> ` replace `</s>]` with ` </bgcolor></fgcolor>`)" displays all available columns to be filtered with different background/foreground colors including the number of items/results
- **all** keyword returns the list of all columns (visible or hidden) no matter if the ColumnFilterButton property is True or False. At runtime, the all keyword may return a string such as "<fgcolor=C0C0C0>[<s>OrderID</s>]</fgcolor><fgcolor> </fgcolor> [EmployeeID] = '4| 5| 6'<fgcolor> </fgcolor><fgcolor=C0C0C0> [<s>OrderDate</s>]</fgcolor><fgcolor> </fgcolor><fgcolor=C0C0C0> [<s>RequiredDate</s>]</fgcolor><fgcolor>", so the control automatically applies HTML format, which you can change it. For instance, "all", displays the current filter, including all other columns. For instance, the "((all + `<fgcolor=808080>` + (matchitemcount < 0 ? ((len(allui) ? `` : ``) + `<r>` + abs(matchitemcount + 1) + ` result(s)`) : (`<r><fgcolor=808080>`+ itemcount + ` item(s)`))) replace `[<b>` with `<bgcolor=000000><fgcolor=FFFFFF><b> ` replace `</b>]` with ` </bgcolor>

</fgcolor>` replace `[<s>` with `<bgcolor=C0C0C0><fgcolor=FFFFFF> ` replace `</s>]` with ` </bgcolor></fgcolor>`)" displays all columns with different background/foreground colors including the number of items/results

Also, the FilterBarCaption property supports predefined constants and operators/functions as described here.

Also, the FilterBarCaption 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 ExFileView.FilterBarDropDownHeight as Double

Specifies the height of the drop down filter window proportionally with the height of the control's list.

Type	Description
Double	A double expression that indicates the height of the drop down filter window. The meaning of the value is explained bellow.

By default, the FilterBarDropDownHeight property is 0.5. It means, the height of the drop down filter window is half of the height of the control's list. Use the FilterBarDropDownHeight property to specify the height of the drop down window filter window. Use the ColumnFilterButton property to display a filter button to the column's caption. Use the Description property to define predefined strings in the filter bar. Use the FilterInclude property to specify whether the child items should be included to the list when the user applies the filter.

If the FilterBarDropDownHeight property is negative, the absolute value of the FilterBarDropDownHeight property indicates the height of the drop down filter window in pixels. In this case, the height of the drop down filter window is not proportionally with the height of the control's list area. For instance, the following sample specifies the height of the drop down filter window being 100 pixels:

```
With ExFileView1
   .FilterBarDropDownHeight = -100
End With
```

If the FilterBarDropDownHeight property is greater than 0, it indicates the height of the drop down filter window proportionally with the height of the control's height list. For instance, the following sample specifies the height of the drop down filter window being the same with the height of the control's list area:

```
With ExFileView1
   .FilterBarDropDownHeight = 1
End With
```

The drop down filter window always include an item.

property ExFileView.FilterBarDropDownWidth(ColumnName as String) as Double

Specifies the width of the drop down filter window proportionally with the width of the control's column. /*not supported in the lite version*/

Type	Description
ColumnName as String	A string expression that indicates the column's name. The valid values are: 'Name', 'Size', 'Modified', 'Type', 'In Folder'
Double	A double expression that indicates the width of the drop down filter window.

Use the FilterBarDropDownWidth property to specify the width of the drop down window filter window. Use the ColumnFilterButton property to display a filter button to the column's caption. BY default, the FilterBarDropDownWidth property is 1. It means, the width of the drop down filter window is the same with the width of the column.

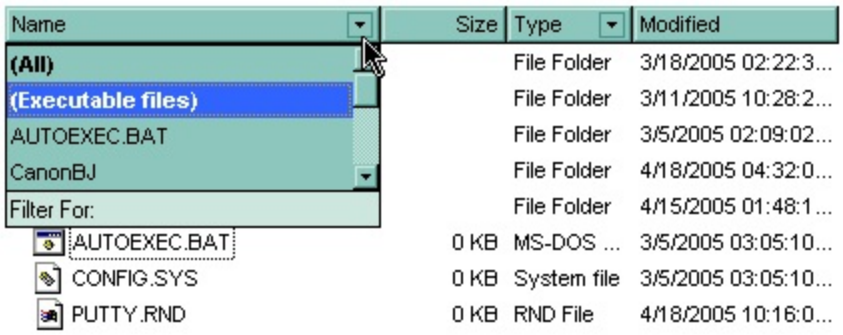
If the FilterBarDropDownWidth property is negative, the absolute value of the FilterBarDropDownWidth property indicates the width of the drop down filter window in pixels. In this case, the width of the drop down filter window is not proportionally with the width of the column. For instance, the following sample specifies the width of the drop down filter window being 100 pixels:

```
With ExFileView1
   .FilterBarDropDownWidth("Name") = -100
End With
```

If the FilterBarDropDownWidth property is greater than 0, it indicates the width of the drop down filter window proportionally with the width of column. For instance, the following sample specifies the width of the drop down filter window being half of the width of the column:

```
With ExFileView1
   .FilterBarDropDownWidth("Name") = 0.5
End With
```

The drop down filter window always include an item.



The following VB sample adds custom filter patterns for executable files:

```
With ExFileView1
   .FilterBarDropDownHeight = 0.7
   .ColumnFilterButton("Name") = True
   .AddColumnCustomFilter "Name", "(Executable files)", "*.exe|*.com|*.bat"
End With
```

The following C++ sample adds custom filter patterns for executable files:

```
m_fileview.SetFilterBarDropDownHeight( 0.7 );
m_fileview.SetColumnFilterButton("Name", TRUE );
m_fileview.AddColumnCustomFilter("Name", "(Executable files)", "*.exe|*.com|*.bat");
```

The following VB.NET sample adds custom filter patterns for executable files:

```
With AxExFileView1
   .FilterBarDropDownHeight = 0.7
   .set_ColumnFilterButton("Name", True)
   .AddColumnCustomFilter("Name", "(Executable files)", "*.exe|*.com|*.bat")
End With
```

The following C# sample adds custom filter patterns for executable files:

```
axExFileView1.FilterBarDropDownHeight = 0.7;
axExFileView1.set_ColumnFilterButton("Name", true);
axExFileView1.AddColumnCustomFilter("Name", "(Executable files)", "*.exe|*.com|*.bat");
```

The following VFP sample adds custom filter patterns for executable files:

```
With thisform.ExFileView1
   .FilterBarDropDownHeight = 0.7
   .Object.ColumnFilterButton("Name") = .t.
```

```
    .AddColumnCustomFilter("Name", "(Executable files)", "*.exe|*.com|*.bat")
EndWith
```

property ExFileView.FilterBarFont as IFontDisp

Retrieves or sets the font for control's filter bar.

Type	Description
IFontDisp	A font object that indicates the font used to paint the description for control's filter

Use the FilterBarFont property to specify the font for the control's filter bar object. Use the Font property to set the control's font. Use the FilterBarHeight property to specify the height of the filter bar. Use the FilterBarCaption property to define the control's filter bar caption. Use the Refresh method to refresh the control.

property ExFileView.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.

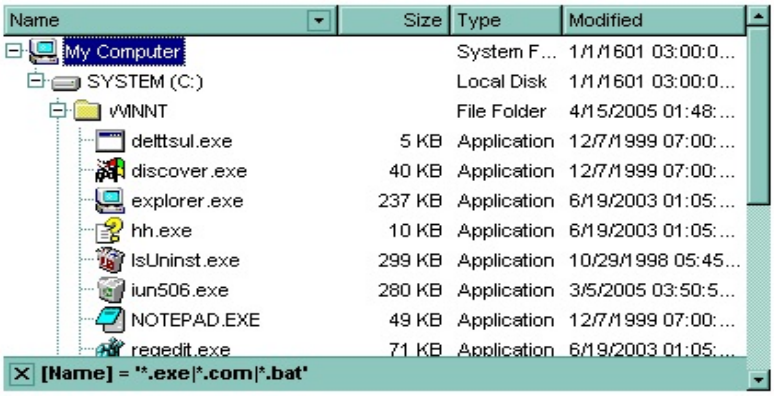
Use the FilterBarForeColor and FilterBarBackColor properties to define colors used to paint the description of the control's filter. Use the FilterBarFont property to specify the filter bar's font. Use the FilterBarCaption property to specify the caption of the control's filter bar.

property ExFileView.FilterBarHeight as Long

Specifies the height of the control's filter bar. If the value is less than 0, the filter bar is automatically resized to fit its description.

Type	Description
Long	A long expression that indicates the height of the filter bar status.

The filter bar status defines the control's filter description. If the FilterBarHeight property is less than 0 the control automatically updates the height of the filter's description to fit in the control's client area. If the FilterBarHeight property is zero the filter's description is hidden. If the FilterBarHeight property is grater than zero it defines the height in pixels of the filter's description. Use the ClearFilter method to clear the control's filter. Use the FilterBarCaption property to define the control's filter bar caption. Use the FilterBarFont property to specify the font for the control's filter bar. Use the FilterBarDropDownHeight to specify the height of the drop down filter window.



property ExFileView.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 FilterBarPromptColumns property specifies the list of columns to be used when filtering by prompt. The ColumnFilterButton property specifies whether the column's header displays a filter button. The control fires the FilterChange once the list gets filtered. Use the FilterBarCaption property to change the caption in the filter bar once a new filter is applied. The FilterBarFont property specifies the font to be used in the filter bar. 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

The FilterBarPrompt property has effect only if:

- FilterBarPromptVisible property is True
- FilterBarPromptPattern property is Empty.

property ExFileView.FilterBarPromptColumns as Variant

Specifies the list of columns to be used when filtering using the prompt.

Type	Description
Variant	A long expression that indicates the index of the column to apply the filter prompt, a string expression that specifies the list of columns (indexes) separated by comma to apply the filter prompt, or a safe array of long expression that specifies the indexes of the columns to apply the filter. 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.

By default, the FilterBarPromptColumns property is -1. If the FilterBarPromptColumns property is -1, the filter prompt is applied for all columns, visible or hidden. Use the FilterBarPromptColumns property to specify the list of columns to apply the filter prompt pattern. The FilterBarPromptVisible property specifies whether the filter prompt is visible or hidden. 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. Changing the FilterBarPromptPattern property does not require calling the ApplyFilter method to apply the new filter, only if filtering is required right a way. The FilterBarPromptType property specifies the type of filtering when the user edits the prompt in the filter bar.

property ExFileView.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. If the FilterBarPromptPattern property is empty, the filter bar displays the FilterBarPrompt property, if the FilterBarPromptVisible property is True. The FilterBarPromptPattern property indicates the patter to filter the list. The pattern may include wild characters if the FilterBarPromptType property is exFilterPromptPattern. The FilterBarPromptColumns specifies the list of columns to be used when filtering. Changing the FilterBarPromptPattern property does not require calling the ApplyFilter method to apply the new filter, only if filtering is required right a way.

property ExFileView.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 FilterBarPromptVisible property to show the filter prompt. 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. Changing the FilterBarPromptPattern property does not require calling the ApplyFilter method to apply the new filter, only if filtering is required right a way. The FilterBarPromptColumns property specifies the list of columns to be used when filtering by prompt. The ColumnFilterButton property specifies whether the column's header displays a filter button. The control fires the FilterChange once the list gets filtered. Use the FilterBarCaption property to change the caption in the filter bar once a new filter is applied.

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 ExFileView.FilterBarPromptVisible as FilterBarVisibleEnum

Shows or hides the control's filter bar including filter prompt.

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

By default, The FilterBarPromptVisible property is exFilterBarHidden. 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 FilterBarPromptVisible property to show the filter prompt. 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. Changing the FilterBarPromptPattern property does not require calling the ApplyFilter method to apply the new filter, only if filtering is required right a way. The FilterBarCaption property defines the caption to be displayed on the control's filter bar. The FilterBarPromptType property specifies the type of filtering when the user edits the prompt in the filter bar. The FilterBarPromptColumns property specifies the list of columns to be used when filtering by prompt. The ColumnFilterButton property specifies whether the column's header displays a filter button. The control fires the FilterChange once the list gets filtered.

The following screen show shows the filter prompt:

The following screen show shows the list once the user types "london":

Name	Title	City
Steven Buchanan	Sales Manager	London
Michael Suyama	Sales Representative	London
Robert King	Sales Representative	London
Anne Dodsworth	Sales Representative	London

london

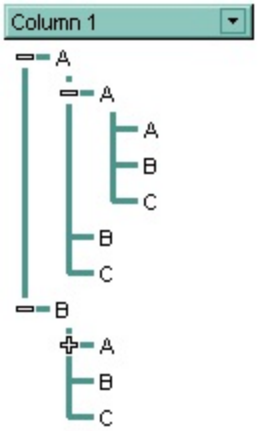
property ExFileView.FilterInclude as FilterIncludeEnum

Specifies the items being included after the user applies the filter. /*not supported in the lite version*/

Type	Description
FilterIncludeEnum	A FilterIncludeEnum expression that indicates the items being included when the filter is applied.

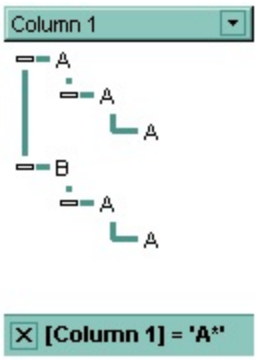
By default, the FilterInclude property is exItemsWithoutChilds. Use the FilterInclude property to specify whether the child items should be included to the list when the user applies the filter. Use the ColumnFilter property and ColumnFilterType property to specify the column's filter. Use the ApplyFilter to apply the filter at runtime. Use the ClearFilter method to clear the control's filter. Use the ExpandFolders property to include child folders in the list.

Let's say that we have the following hierarchy:

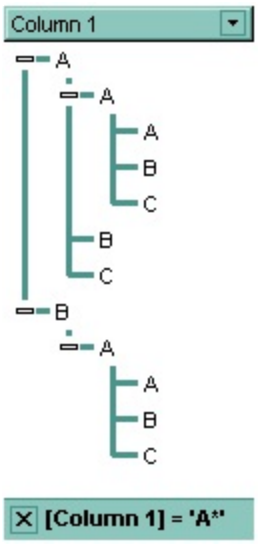


and the ColumnFilter property is "A*", ColumnFilterType property is FilterPattern.

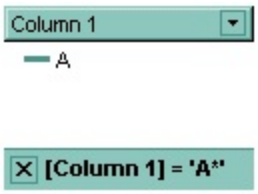
If the FilterInclude property is **exItemsWithoutChilds**, the filtered list looks like follows:



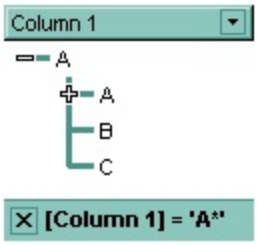
If the FilterInclude property is **exItemsWithChilds**, the filtered list looks like follows:



If the FilterInclude property is **exRootsWithoutChilds**, the filtered list looks like follows:



If the FilterInclude property is **exRootsWithChilds**, the filtered list looks like follows:



property ExFileView.Font as IFontDisp

Retrieves or sets the Font object used to paint control.

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

Use the Font property to change the control's font. Use the Bold property to bold files that matches a pattern. Use the Bold property to bold a specified file or folder. Use the Refresh method to refresh the control.

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

```
With ExFileView1
   With .Font
      .Name = "Tahoma"
   End With
   .Refresh
End With
```

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

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

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

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

```
With AxExFileView1
   Dim font As System.Drawing.Font = New System.Drawing.Font("Tahoma", 10,
FontStyle.Regular, GraphicsUnit.Point)
   .Font = font
   .CtlRefresh()
End With
```

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

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

```
axExFileView1.Font = font;
axExFileView1.CtlRefresh();
```

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

```
with thisform.ExFileView1.Object
   .Font.Name = "Tahoma"
   .Refresh()
endwith
```

property ExFileView.ForeColor as Color

Retrieves or sets the control's foreground color.

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

Use the ForeColor property to specify the control's foreground color. Use the BackColor property to specify the control's background color. Use the ForeColor property to specify the foreground color for files or folders that match specified patterns. Use the SelForeColor and SelBackColor properties to specify the background and foreground colors for selected items.

property ExFileView.ForeColorHeader as Color

Specifies the header's foreground color.

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

Use the ForeColorHeader properties to specify the foreground color for the control's header bar. Use the BackColorHeader property to specify the background color for the control's header bar. The HeaderVisible property shows or hides the control's header bar. Use the FilterBarForeColor property to specify the foreground color for the control's filter bar. Use the ForeColor property to specify the control's foreground color.

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

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

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

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

For instance:

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

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

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

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

method ExFileView.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.

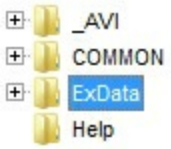
property ExFileView.FullRowSelect as Boolean

Enables full-row selection in the control.

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

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

The following screen shot shows the control with FullRowSelect property on False (by default):



The following screen shot shows the control with FullRowSelect property on True:

property ExFileView.Get (Type as TypeEnum) as Files

Builds and gets the collection of File objects of the given type.

Type	Description
Type as TypeEnum	Use SelItems to get the collection of selected files/folder, or AllItems for retrieving the entire collection of files/folders.
Files	A Files collection that contains files based on the specified type.

Use the HasCheckBox property to assign a check box for each item in your control. Use the BrowseFolderPath property to specify the path to the browsed folder. Use the Name property to specify the name of the file or folder. Use the FullName property to get the full name of the file or folder. Use the Folder property to specify whether the File object hosts a file or a folder.

Use the Get method to retrieve :

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

The File.Children property helps you to collect recursively all files/folders of specified object. The EnumR function displays the full name of each file/folder, and goes recursively to each subfolder. Use the Folder property to specify whether the File object hosts a file or a folder. The Children property returns a collection of File objects, if the ExpandFolders property is True.

```
Public Sub EnumR(ByVal f As EXFILEVIEWLibCtl.File)
   Debug.Print f.FullName
   For Each c In f.Children
      EnumR (c)
   Next
End Sub
```

The following VB sample displays the list of files as they are displayed:

```
With ExFileView1.Get(VisibleItems)
   For i = 0 To .Count - 1
      With .Item(i)
```

```
        Debug.Print .Name
      End With
    Next
End With
```

The following VB enumerates the selected files and folders:

```
Dim i As Long
With ExFileView1.Get(SelItems)
    For i = 0 To .Count - 1
        Debug.Print .Item(i).FullName
    Next
End With
```

The following C++ sample displays the list of files as they are displayed:

```
CFiles files = m_fileview.GetGet( 3 /*VisibleItems*/ );
for ( long i = 0; i < files.GetCount(); i++ )
    OutputDebugString( files.GetItem( COleVariant( i ) ).GetName() );
```

The following C++ enumerates the selected files and folders:

```
CFiles files = m_fileview.GetGet( 0 /*SelItems*/ );
for ( long i = 0; i < files.GetCount(); i++ )
    OutputDebugString( files.GetItem( COleVariant( i ) ).GetFullName() );
```

The following VB.NET sample displays the list of files as they are displayed:

```
With AxExFileView1.get_Get(EXFILEVIEWLib.TypeEnum.VisibleItems)
    Dim i As Integer
    For i = 0 To .Count - 1
        With .Item(i)
            Debug.WriteLine(.Name())
        End With
    Next
End With
```

The following VB.NET enumerates the selected files and folders:

```
With AxExFileView1.get_Get(EXFILEVIEWLib.TypeEnum.SelItems)
```

```
    Dim i As Integer
    For i = 0 To .Count - 1
        Debug.WriteLine(.Item(i).FullName)
    Next
End With
```

The following C# sample displays the list of files as they are displayed:

```
EXFILEVIEWLib.Files files = axExFileView1.get_Get(EXFILEVIEWLib.TypeEnum.VisibleItems);
for (int i = 0; i < files.Count; i++)
{
    EXFILEVIEWLib.File file = files[i];
    System.Diagnostics.Debug.WriteLine(file.Name);
}
```

The following C# enumerates the selected files and folders:

```
EXFILEVIEWLib.Files files = axExFileView1.get_Get(EXFILEVIEWLib.TypeEnum.SelItems);
for (int i = 0; i < files.Count; i++)
{
    EXFILEVIEWLib.File file = files[i];
    System.Diagnostics.Debug.WriteLine(file.FullName);
}
```

The following VFP sample displays the list of files as they are displayed:

```
With thisform.ExFileView1.Get(3)    && VisibleItems
    For i = 0 To .Count - 1
        With .Item(i)
            wait window nowait .Name
        EndWith
    Next
EndWith
```

The following VFP enumerates the selected files and folders:

```
with thisform.ExFileView1.Get( 0 ) && SelItems
    local i
    for i = 0 to .Count - 1
```

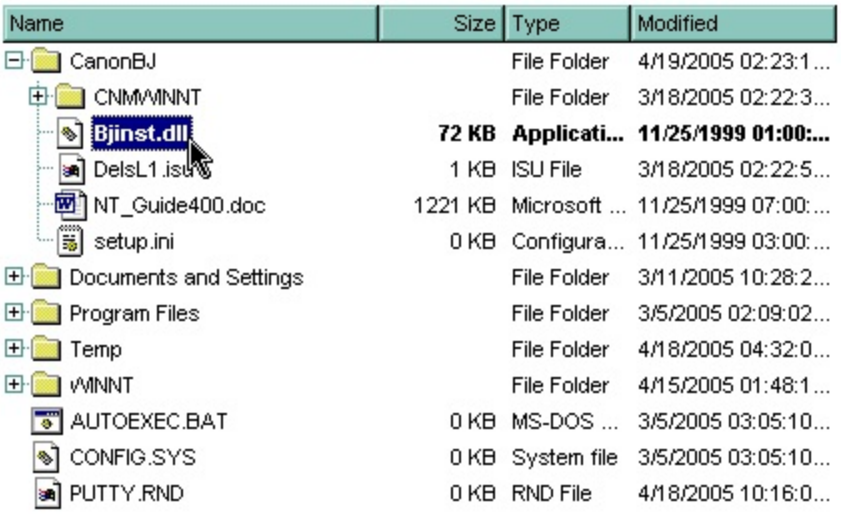
```
    with .Item(i)
        wait window nowait .FullName
    endwith
  next
endwith
```

property ExFileView.HasButtons as Boolean

Adds a button to the left side of each parent item.

Type	Description
Boolean	A boolean expression that indicates whether the control displays a left button for each item that contains child items.

By default, the HasButtons property is True. Use the HasButtons property to specify whether the parent items displays a +/- sign to let user expands or collapses items. The HasLines property retrieves or sets a value that indicates whether the control links the child items to their parents. Use the HasLinesAtRoot property retrieves or sets a value that indicates whether the control draws the lines that link the root items. The HasButtons property has effect only if the ExpandFolders property it True. Use the ExpandOnDblClk property to expand or collapse a folder when user double clicks the folder. Use the ExploreFromHere property specifies the root folder for the control.

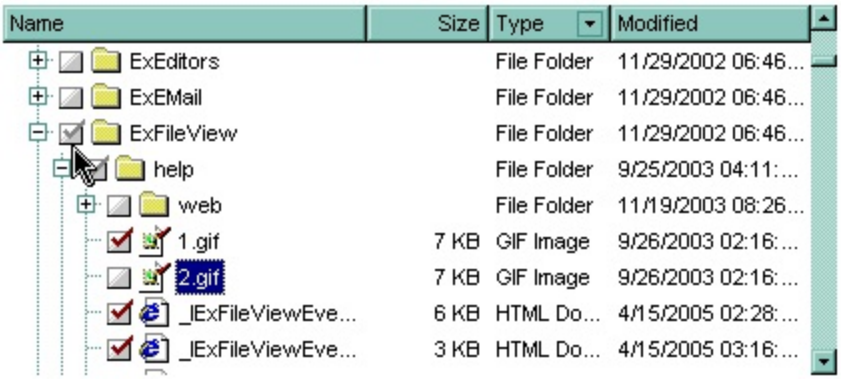


property ExFileView.HasCheckBox as CheckBoxEnum

Specifies whether the control displays a check box for each item. /*not supported in the lite version*/

Type	Description
CheckBoxEnum	A CheckBoxEnum expression that indicates whether the control displays a check box for each item.

Use the HasCheckBox property to assign a check box for each item in your control. The control supports partial check feature (three-states check box) too. When partial check feature is on, the control displays a partial check box for an item that contains checked items as well as unchecked child items. Use the Checked property to check or uncheck by code a file or folder. Use the Get method to get the collection of checked items.



The following VB sample displays the checked files and folders:

```
With ExFileView1.Get(CheckItems)
   Dim i As Long
   For i = 0 To .Count - 1
      With .Item(i)
         Debug.Print .Name
      End With
   Next
End With
```

The following C++ sample displays the checked files and folders:

```
CFiles files = m_fileview.GetGet( 2 /*CheckItems*/ );
for ( long i = 0; i < files.GetCount(); i++ )
   OutputDebugString( files.GetItem( COleVariant( i ) ).GetName() );
```

The following VB.NET sample displays the checked files and folders:

```
With AxExFileView1.get_Get(EXFILEVIEWLib.TypeEnum.CheckItems)
    Dim i As Integer
    For i = 0 To .Count - 1
        Debug.WriteLine(.Item(i).Name)
    Next
End With
```

The following C# sample displays the checked files and folders:

```
EXFILEVIEWLib.Files files = axExFileView1.get_Get(EXFILEVIEWLib.TypeEnum.CheckItems);
for (int i = 0; i < files.Count; i++)
    System.Diagnostics.Debug.WriteLine( files[i].Name );
```

The following VFP sample displays the checked files and folders:

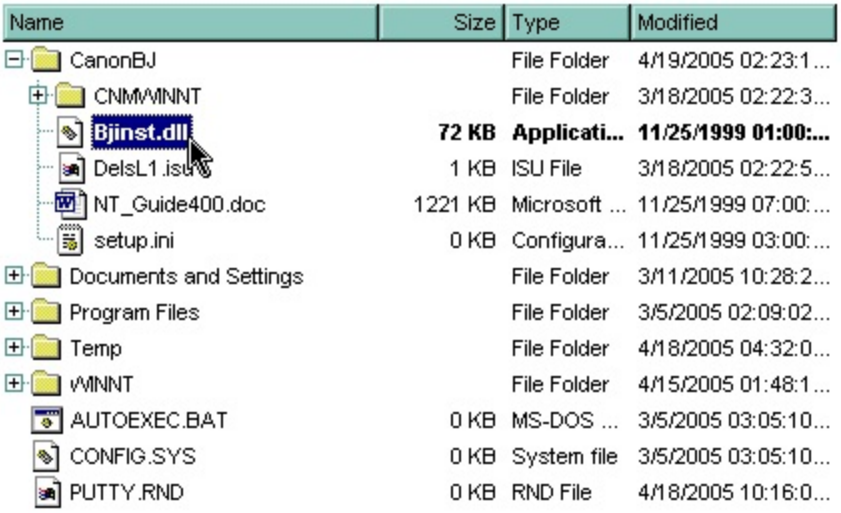
```
With thisform.ExFileView1.Get( 2 ) && CheckItems
    local i
    for i = 0 to .Count - 1
        with .Item(i)
            wait window nowait .Name
        endwith
    next
EndWith
```

property ExFileView.HasLines as Boolean

Retrieves or sets a value that indicates whether the control links the child items to their parents.

Type	Description
Boolean	A Boolean expression that indicates whether the control links the child items to their parents.

By default, the HasLines property is True. The HasLines property retrieves or sets a value that indicates whether the control links the child items to their parents. Use the HasButtons property to specify whether the parent items displays a +/- sign to let user expands or collapses items. Use the HasLinesAtRoot property retrieves or sets a value that indicates whether the control draws the lines that link the root items. The HasLines property has effect only if the ExpandFolders property it True.

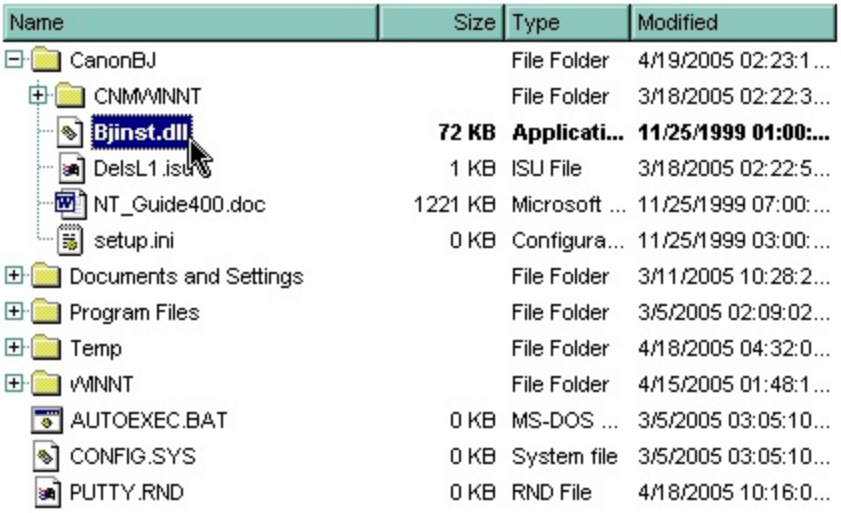


property ExFileView.HasLinesAtRoot as Boolean

Retrieves or sets a value that indicates whether the control draws the lines that link the root items.

Type	Description
Boolean	A boolean expression that indicates whether the control draws the lines that link the root items.

By default, the HasLinesAtRoot property is False. Use the HasLinesAtRoot property retrieves or sets a value that indicates whether the control draws the lines that link the root items. The HasLines property retrieves or sets a value that indicates whether the control links the child items to their parents. Use the HasButtons property to specify whether the parent items displays a +/- sign to let user expands or collapses items. The HasLinesAtRoot property has effect only if the ExpandFolders property it True.



property ExFileView.HeaderAppearance as AppearanceEnum

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

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

Use the HeaderAppearance property to change the appearance for the control's header bar. Use the HeaderVisible property to hide the control's header bar. Use the HeaderHeight property to change the height of the control's header bar. Use the ColumnVisible property to hide or show a column. Use the ColumnCaption property to specify the column's caption. Use the ColumnWidth property to change the column's width. Use the BackColorHeader property to specify the background color for the control's header bar. Use the ForeColorHeader property to specify the foreground color for the control's header bar.

property ExFileView.HeaderHeight as Long

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

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

By default, the HeaderHeight property is 18 pixels. Use the HeaderHeight property to change the height of the control's header bar. Use the HeaderVisible property to hide the control's header bar. The HeaderAppearance property changes the appearance for the control's header bar.

property ExFileView.HeaderVisible as Boolean

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

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

Use the HeaderVisible property to hide the control's header bar. Use the HeaderHeight property to change the height of the control's header bar. Use the ColumnVisible property to hide or show a column. Use the ColumnCaption property to specify the column's caption. Use the ColumnWidth property to change the column's width. Use the BackColorHeader property to specify the background color for the control's header bar. Use the ForeColorHeader property to specify the foreground color for the control's header bar. Use the FilterBarHeight property to specify the height of the control's filter bar. Use the BackColor property to specify the control's background color.

property ExFileView.HideSelection as Boolean

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

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

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

property ExFileView.HotBackColor as Color

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

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

By default, the HotBackColor property is 0, which means that the HotBackColor property has no effect. Use the HotBackColor property on a non-zero value to highlight the item from the cursor. The HotForeColor property specifies the foreground color to highlight the item from the cursor. The SelBackColor property specifies the selection background color.

property ExFileView.HotForeColor as Color

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

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

By default, the HotForeColor property is 0, which means that the HotForeColor property has no effect. Use the HotForeColor property on a non-zero value to highlight the item from the cursor. The HotBackColor property specifies the background color to highlight the item from the cursor. The SelForeColor property specifies the selection foreground color.

property ExFileView.hWnd as Long

Retrieves the control's window handle.

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

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

property ExFileView.ImageSize as Long

Retrieves or sets the size of icons the control displays.

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

By default, the ImageSize property is 16 (pixels). The ImageSize property specifies the size of the icons the control displays.

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

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

property ExFileView.IncludeFiles as Boolean

Retrieves or sets a value indicating whether the control includes the files to the list.

Type	Description
Boolean	A boolean expression indicating whether the control includes the files to the list.

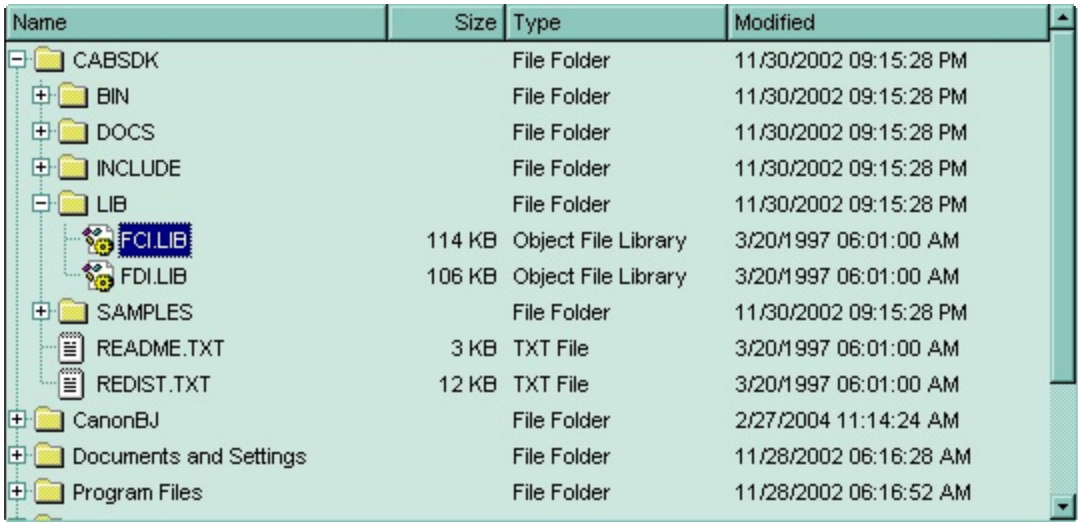
Use the IncludeFiles property to include files in the current list. Use the IncludeFilesInFolder property to include files when expanding a folder. Use the IncludeFilter property to specify the patterns for files being included in the list. Use the ExcludeFilter to exclude files from the current list. Use the IncludeFolders property to include folders in the current list. The ExpandFolders property retrieves or sets a value that indicates whether the control expands the folder objects. Use the ExploreFromHere property to specify the root folder for the control.

property ExFileView.IncludeFilesInFolder as Boolean

Retrieves or sets a value that indicates whether the control includes files when expanding a folder.

Type	Description
Boolean	A boolean expression indicates whether the control includes the files when expanding a folder.

Use the IncludeFilesInFolder property to include files when expanding a folder. The IncludeFilesInFolder property has effect only if the IncludeFiles property is True, and ExpandFolders property is True. The ExpandFolders property retrieves or sets a value that indicates whether the control expands the folder objects. The IncludeFilter and ExcludeFilter properties filters the files being included in the list. Use the ExpandOnDblClk property to expand or collapse a folder when user double clicks the folder. Use the ExploreFromHere property specifies the root folder for the control.

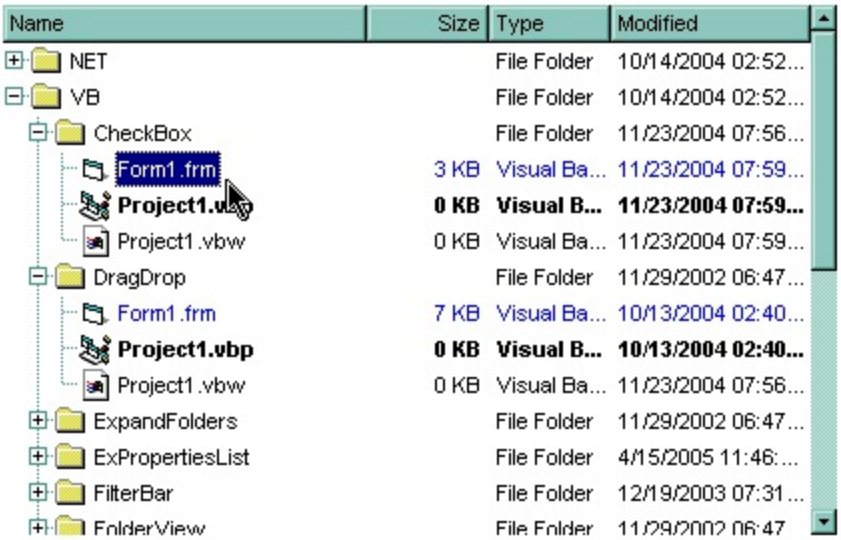


property ExFileView.IncludeFilter as String

Specifies the pattern used to include files to the control's list, like '*.cpp *.h'

Type	Description
String	A string expression that may contain wild cards like * or ?.

Use the IncludeFilter property to include files that match a pattern. When the IncludeFilter is called, the control automatically refreshes the current list, and applies the FileType attributes. To remove a previous include filter you can use "" empty string. By default, the IncludeFilter is "". Use the ExcludeFilter to exclude files from the current list. The IncludeFolderFilter property specifies a wild characters expression that indicates the folders being included. The ExcludeFolderFilter property specifies a wild characters expression that indicates the folders being excluded. The patterns are separated by space character. Use the IncludeFiles property to include files in the control's list. Use the exHideFileExtensionsForKnownFileTypes option to show the file extensions in case your Windows Explorer, the "**Hide File Extensions For Known File Types**" is checked.



The following VB sample includes only the files with the extensions: 'frm', 'frx', 'vbp' and 'vbw', and colorize the items differentially:

```
With ExFileView1
   .IncludeFilter = "*.frm *.frx *.vbp *.vbw"
   With .FileTypes
      With .Add("*.frm *.frx")
         .ForeColor = vbBlue
      End With
      With .Add("*.vbp")
         .Bold = True
```

```
        End With
        .Apply
    End With
End With
```

The following C++ sample includes only the files with the extensions: 'frm', 'frx', 'vbp' and 'vbw', and colorize the items differentially:

```
m_fileview.SetIncludeFilter("*.frm *.frx *.vbp *.vbw");
CFileTypes fileTypes = m_fileview.GetFileTypes();
fileTypes.Add( "*.frm *.frx" ).SetForeColor( RGB(0,0,255) );
fileTypes.Add( "*.vbp" ).SetBold( TRUE );
fileTypes.Apply();
```

The following VB.NET sample includes only the files with the extensions: 'frm', 'frx', 'vbp' and 'vbw', and colorize the items differentially:

```
With AxExFileView1
    .IncludeFilter = "*.frm *.frx *.vbp *.vbw"
    With .FileTypes
        With .Add("*.frm *.frx")
            .ForeColor = ToUInt32(Color.Blue)
        End With
        With .Add("*.vbp")
            .Bold = True
        End With
        .Apply()
    End With
End With
```

where the ToUInt32 function converts a Color expression to an OLE_COLOR type,

```
Shared Function ToUInt32(ByVal c As Color) As UInt32
    Dim i As Long
    i = c.R
    i = i + 256 * c.G
    i = i + 256 * 256 * c.B
    ToUInt32 = Convert.ToUInt32(i)
End Function
```

The following C# sample includes only the files with the extensions: 'frm', 'frx', 'vbp' and 'vbw', and colorize the items differentially:

```
axExFileView1.IncludeFilter = "*.frm *.frx *.vbp *.vbw";
EXFILEVIEWLib.FileTypes fileTypes = axExFileView1.FileTypes;
fileTypes.Add("*.frm *.frx").ForeColor = ToUInt32(Color.Blue);
fileTypes.Add("*.vbp").Bold = true;
fileTypes.Apply();
```

where the ToUInt32 function converts a Color expression to an OLE_COLOR type,

```
private UInt32 ToUInt32(Color c)
{
   long i;
   i = c.R;
   i = i + 256 * c.G;
   i = i + 256 * 256 * c.B;
   return Convert.ToUInt32(i);
}
```

The following VFP sample includes only the files with the extensions: 'frm', 'frx', 'vbp' and 'vbw', and colorize the items differentially:

```
With thisform.ExFileView1
   .IncludeFilter = "*.frm *.frx *.vbp *.vbw"
   With .FileTypes
      With .Add("*.frm *.frx")
         .ForeColor = RGB(0,0,255)
      EndWith
      With .Add("*.vbp")
         .Bold = .t.
      EndWith
      .Apply
   EndWith
EndWith
```

property ExFileView.IncludeFolderFilter as String

Retrieves or sets a value that indicates the folders being included.

Type	Description
String	A string expression that may contain wild cards like * or ?.

Use the IncludeFolderFilter property to include folders that match a pattern or a list of patterns. When the IncludeFolderFilter is invoked, the control automatically refreshes the current list, and applies the FileType attributes. The IncludeFolderFilter property has effect if it is not empty. Use the IncludeFilter property to include files that match a pattern or a list of patterns. The ExcludeFolderFilter property specifies a wild characters expression that indicates the folders being excluded. Use the ExcludeFilter to exclude files from the current list. Use the IncludeFiles property to include files in the control's list. Use the exHideFileExtensionsForKnownFileTypes option to show the file extensions in case your Windows Explorer, the "**Hide File Extensions For Known File Types**" is checked.

The following VB sample includes the "Temp" folders:

```
With ExFileView1
   .IncludeFolderFilter = "*temp*"
End With
```

The following C++ sample includes the "Temp" folders:

```
m_fileview.SetIncludeFolderFilter( "temp*" );
```

The following VB.NET sample includes the "Temp" folders:

```
AxExFileView1.IncludeFolderFilter = "*temp*"
```

The following C# sample includes the "Temp" folders:

```
axExFileView1.IncludeFolderFilter = "*temp*";
```

The following VFP sample includes the "Temp" folders:

```
With thisform.ExFileView1
   .IncludeFolderFilter = "*temp*"
EndWith
```

property ExFileView.IncludeFolders as Boolean

Retrieves or sets a value that indicates whether control includes the folders.

Type	Description
Boolean	A boolean expression that indicates whether control includes the folders.

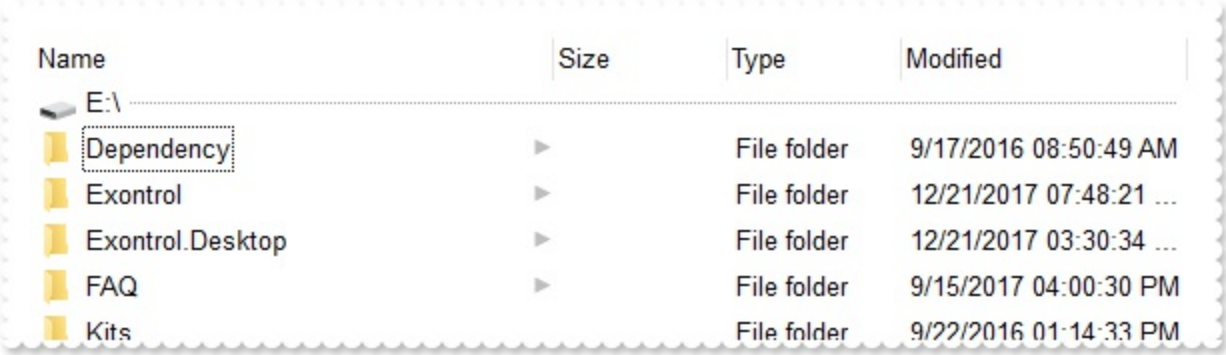
The IncludeFolders property specifies whether the control shows the folders. The ExpandFolders property retrieves or sets a value that indicates whether the control expands the folder objects. The Folder property specifies whether a File object holds a file or a folder. Use the ExploreFromHere property specifies the root folder for the control. Use the IncludeFilesInFolder property to include files when expanding a folder. Use the IncludeFiles property to include files in the current list. The IncludeFolderFilter property specifies a wild characters expression that indicates the folders being included. The ExcludeFolderFilter property specifies a wild characters expression that indicates the folders being excluded.

property ExFileView.IncludeParent as IncludeParentEnum

Retrieves or sets a value that indicates whether the control includes the parent folder.

Type	Description
IncludeParentEnum	An IncludeParentEnum expression that indicates whether the control includes the parent folder.

If the IncludeParent property is True, the control adds a new item titled ".." that defines the parent folder for the browsed folder. If the user clicks on this item, the control browses the parent folder. The BrowseFolderPath property specifies the path to the browsed folder. The control fires the StateChange event when the user changes the browsed path. The ExploreFromHere property specifies the root folder for the control. Use the IncludeParentIconKey property to specify the key of the icon being displayed for parent folders. Use the Folder property to specify whether a File object holds a file or a folder. Use the IncludeParentLabel property to specify a different label for the parent item.



property ExFileView.IncludeParentIconKey as Long

Retrieves or sets a value that indicates the key of the icon used for 'Parent' button.

Type	Description
Long	A long expression that indicates the index of the icon being displayed for the Parent item.

By default, the IncludeParentIconKey property is -1. Use the LoadIcon property to load icons to the control. If the icon doesn't exist the control displays no icon for Parent item. Use the IncludeParent property to specify whether the browsed folder includes a link to the parent folder. If the IncludeParent property is True, the control adds a new item titled ".." that defines the parent folder for the browsed folder. If the user clicks on this item, the control browses the parent folder. The BrowseFolderPath property specifies the path to the browsed folder. The control fires the StateChange event when the user changes the browsed path.

property ExFileView.IncludeParentLabel as String

Specifies the label for the parent item.

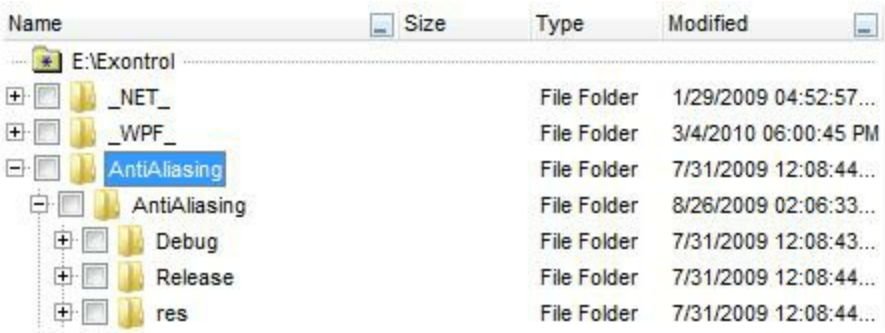
Type	Description
String	A String expression that specifies the label to be displayed in the parent item. The expression may include <%0%>, <%1%>, <%2%> or <%3%>, which indicates the full name, name , parsed name, or relative name of the parent folder.

By default, the IncludeParentLabel property is "..". The IncludeParent property indicates whether the parent folder is being displayed. For instance, the IncludeParentLabel = " <%0%>" specifies that the parent item should display the full name (including its path) of the parent folder. The BrowseFolderPath property indicates the path of browsed folder.

The IncludeParentLabel property supports the following fields:

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

The following screen shot shows the browsed folder (E:\Exontrol):

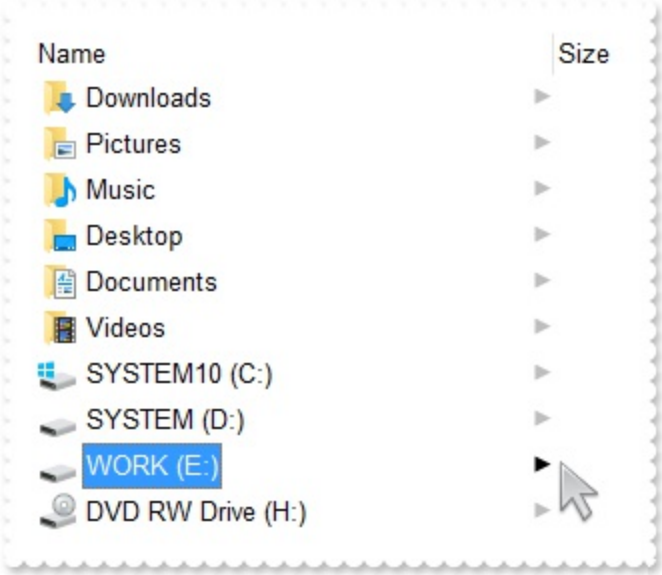


property ExFileView.IncludeSubFolderIconKey as Long

Retrieves or sets a value that indicates the key of the icon to highlights folders that includes sub-folders.

Type	Description
Long	A Long expression that specifies the key of the icon to highlights folders that includes sub-folders.

By default, the IncludeSubFolderIconKey property is -1, which indicates that the control displays an arrow when the folder contains sub-folders as can seen in the following screen shot:



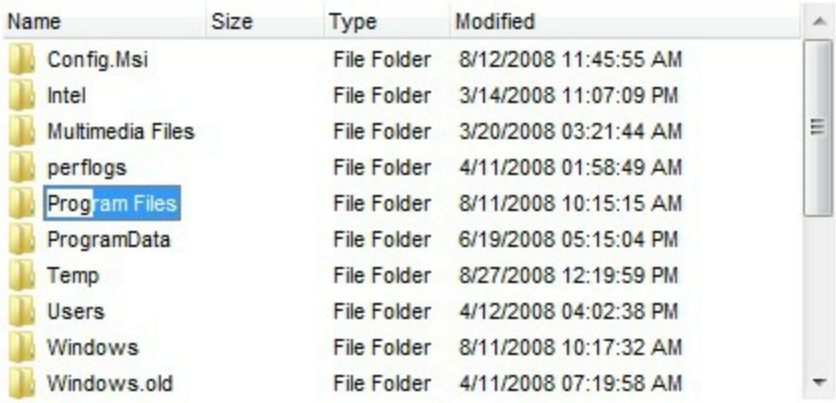
For instance, if IncludeSubFolderIconKey property is 0, the control displays no default arrow for folders that contains sub-folders.

property ExFileView.IncrementalSearch as IncrementalSearchEnum

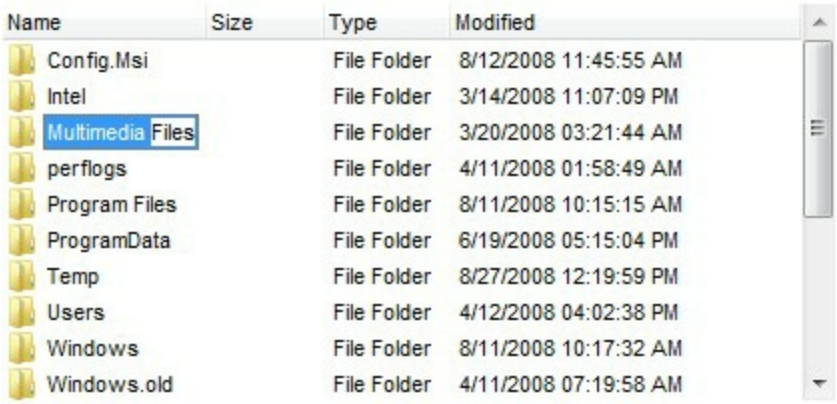
Specifies how the control searches for the objects while user types characters.

Type	Description
IncrementalSearchEnum	An IncrementalSearchEnum expression that specifies how the control searches for objects while tying characters.

By default, the IncrementalSearch property is exDefaultStartWith, and it is similar with incremental searching in the Windows Explorer. An incremental search begins searching as soon as you type the first character of the search string. So, typing characters starts searching for objects. If no character is typed during one second, the incremental search ends. If the IncrementalSearch property is exStartWith or exContains, the control highlights the found characters, while typing as shown in the following screen shots. The exStartWith option searches for objects that starts with typed characters while, the exContains option searches for objects that contains typed characters. Once an object is found and highlighted you can press F3 key for searching the next occurence.



The picture shows the Prog word as being highlighted after the user typed "prog" characters. In this case the IncrementalSearch property is exStartWith



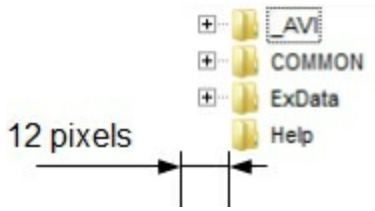
The picture shows the Files word as being highlighted after the user typed "files" characters. In this case the IncrementalSearch property is exContains

property ExFileView.Indent as Long

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

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

By default, the Indent property is 12 pixels. Use the Indent property to increase or decrease the the amount, in pixels, that child items are indented relative to their parent items. The ExpandFolders property specifies whether the folders shows their sub-folders in the same list, with indention.



property ExFileView.IsBusy as Boolean

Indicates whether the control still collects information about current files and folders.

Type	Description
Boolean	A Boolean expression that specifies whether the control is busy or ready.

The IsBusy property indicates whether the control still loading information about files and folders. The control fires the StateChange(ReadyState) once all information on current view is loaded or done. The StateChange(BusyState) event notifies your application once the control start loading information for current files and folders. The information could be size, dates, and icons.

property ExFileView.Layout as String

Saves or loads the control's layout, such as positions of the columns, scroll position, filtering values.

Type	Description
String	A String expression that specifies the control's layout.

You can use the Layout property to store the control's layout and to restore the layout later. For instance, you can save the control's Layout property to a file when the application is closing, and you can restore the control's layout when the application is loaded. The Layout property saves almost all of the control's properties that user can change at runtime (like changing the column's position by drag and drop). The Layout property does NOT save the control's data, so the Layout property should be called once you loaded the data from your database, xml or any other alternative. Once the data is loaded, you can call the Layout property to restore the View as it was saved. Before closing the application, you can call the Layout property and save the content to a file for reading next time the application is opened.

The Layout property saves/loads the following information:

- columns size and position
- current selection
- scrolling position and size
- sorting columns
- expanded/collapsed items, if any
- BrowseFolderPath property

These properties are serialized to a string and encoded in BASE64 format.

The following movies show how Layout works:

- The Layout property is used to save and restore the control's view.

Generally, the Layout property can be used to save / load the control's layout (or as it is displayed). Thought, you can benefit of this property to sort the control using one or more columns as follows:

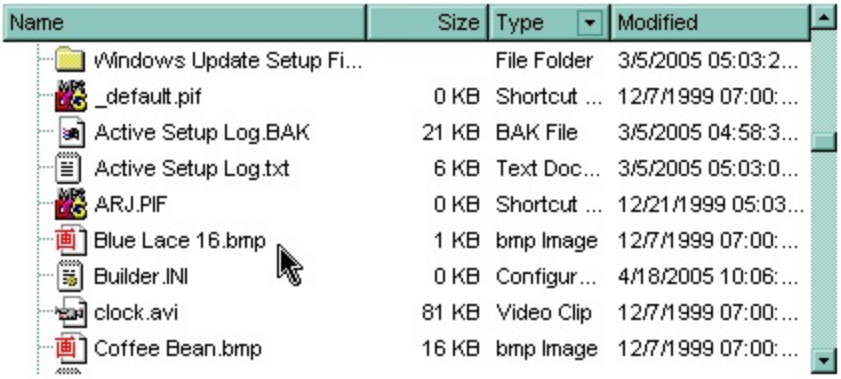
- singlesort="C0:2", sorts descending the name column (index is 0)

method ExFileView.LoadIcon (Icon as Long, IconKey as Long)

Appends a new icon image to control images collection.

Type	Description
Icon as Long	A long expression that indicates the handle of the icon being added.
IconKey as Long	A long expression that indicates the icon's key used by the IconIndex property.

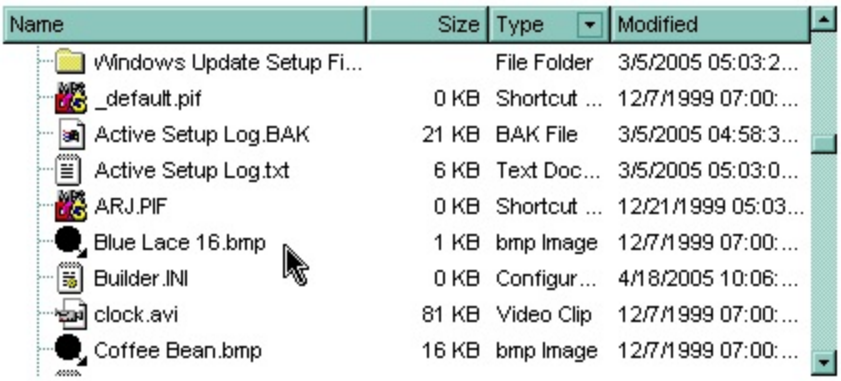
Use the LoadIcon to add an icon to the control images collection. Use the LoadIcons method to load multiple icons. The images collection is used to replace the default file/folder's icon. Use the IconIndex property of FileType object to change the file's icon. Use the BrowseFolderPath property to specify the path to the browsed folder.



The following VB sample replaces the default icon for files of BMP and JPG types with the icon:

```
With ExFileView1
   .LoadIcon LoadPicture("C:\Temp\sample.ico").Handle, 1234
   With .FileTypes.Add("*.bmp *.jpg")
      .IconIndex = 1234
      .Apply
   End With
End With
```

After running the sample the default icons for BMP and JPG files is changed like:



The following C++ sample replaces the default icon for files of BMP and JPG types:

```
IPictureDisp* pPicture = NULL;
if ( LoadPicture( "c:\\temp\\sample.ico", &pPicture ) )
{
   OLE_HANDLE hIcon = NULL;
   if ( CComQIPtr<IPicture> spPicture( pPicture ) )
      spPicture->get_Handle( &hIcon );
   m_fileview.LoadIcon( hIcon, 1234 );

   CFileType fileType = m_fileview.GetFileTypes().Add("*.bmp *.jpg");
   fileType.SetIconIndex( 1234 );
   fileType.Apply();
}
```

where the LoadPicture function loads a picture from a file, and gets the IPictureDisp interface:

```
#include
BOOL LoadPicture( LPCTSTR szFileName, IPictureDisp** ppPictureDisp )
{
   BOOL bResult = FALSE;
   if ( szFileName )
   {
      OFSTRUCT of;
      HANDLE hFile = NULL;;
#ifdef _UNICODE
      USES_CONVERSION;
      if ( (hFile = (HANDLE)OpenFile( W2A(szFileName), &of;, OF_READ |
OF_SHARE_COMPAT)) != (HANDLE)HFILE_ERROR )
```

```
#else
      if ( (hFile = (HANDLE)OpenFile( szFileName, &of;, OF_READ | OF_SHARE_COMPAT)) !=
(HANDLE)HFILE_ERROR )
#endif
      {
         *ppPictureDisp = NULL;
         DWORD dwHighWord = NULL, dwSizeLow = GetFileSize( hFile, &dwHighWord; );
         DWORD dwFileSize = dwSizeLow;
         HRESULT hResult = NULL;
         if ( HGLOBAL hGlobal = GlobalAlloc(GMEM_MOVEABLE, dwFileSize) )
            if ( void* pvData = GlobalLock( hGlobal ) )
            {
               DWORD dwReadBytes = NULL;
               BOOL bRead = ReadFile( hFile, pvData, dwFileSize, &dwReadBytes;, NULL );
               GlobalUnlock( hGlobal );
               if ( bRead )
               {
                  CComPtr spStream;
                  _ASSERTE( dwFileSize == dwReadBytes );
                  if ( SUCCEEDED( CreateStreamOnHGlobal( hGlobal, TRUE, &spStream;) ) )
                     if ( SUCCEEDED( hResult = OleLoadPicture( spStream, 0, FALSE,
IID_IPictureDisp, (void**)ppPictureDisp ) ) )
                        bResult = TRUE;
               }
            }
         CloseHandle( hFile );
      }
   }
   return bResult;
}
```

The following VB.NET sample replaces the default icon for files of BMP and JPG types:

```
With AxExFileView1
   Dim spPicture As stdole.IPictureDisp =
IPDH.GetIPictureDisp(Image.FromFile("c:\temp\sample.ico"))
   .LoadIcon(spPicture.Handle, 1234)
```

```
    With .FileTypes.Add("*.bmp *.jpg")
        .IconIndex = 1234
        .Apply()
    End With
End With
```

where the IPDH class is defined like follows:

```
Public Class IPDH
    Inherits System.Windows.Forms.AxHost

    Sub New()
        MyBase.New("")
    End Sub

    Public Shared Function GetIPictureDisp(ByVal image As Image) As Object
        GetIPictureDisp = AxHost.GetIPictureDispFromPicture(image)
    End Function

End Class
```

The following C# sample replaces the default icon for files of BMP and JPG types:

```
stdole.IPictureDisp spPicture =
IPDH.GetIPictureDisp(Image.FromFile("c:\\temp\\sample.ico")) as stdole.IPictureDisp;
axExFileView1.LoadIcon( spPicture.Handle, 1234);
EXFILEVIEWLib.FileType fileType = axExFileView1.FileTypes.Add("*.bmp *.jpg");
fileType.IconIndex = 1234;
fileType.Apply();
```

where the IPDH class is defined like follows:

```
internal class IPDH : System.Windows.Forms.AxHost
{
    public IPDH() : base("")
    {
    }

    public static object GetIPictureDisp(System.Drawing.Image image)
```

```
    {
        return System.Windows.Forms.AxHost.GetIPictureDispFromPicture( image );
    }
}
```

The following VFP sample replaces the default icon for files of BMP and JPG types:

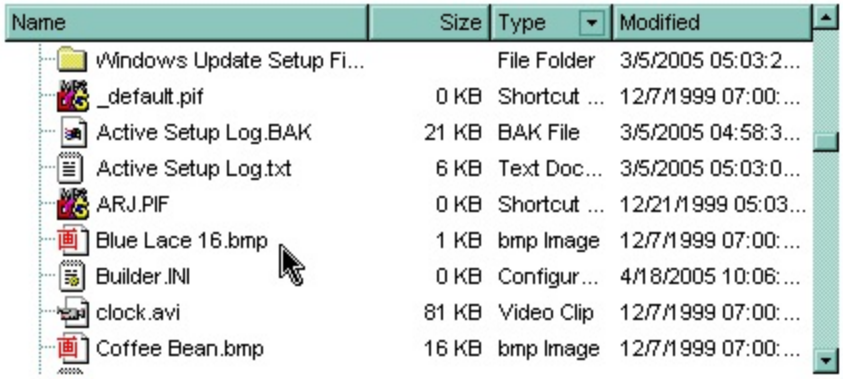
```
With thisform.ExFileView1
    local i
    with LoadPicture("C:\temp\sample.ico")
        i = .Handle()
    endwith
    .Object.LoadIcon(i, 1234)
    With .FileTypes.Add("*.bmp *.jpg")
        .IconIndex = 1234
        .Apply
    EndWith
EndWith
```

method ExFileView.LoadIcons (ImageList as Variant)

Loads new images to the control.

Type	Description
ImageList as Variant	An ImageList control whose images are added to the control's icons collection or a string expression that represents the image list encoded on BASE64 format (uses the eximages tool to encode multiple icons to an image list).

The ImageList control must be provided by the MSCOMCTL.OCX file (Microsoft Windows Common Controls). The ImageList must contains a collection of icon files. The IconIndex must use the the key in the icons collection. Use the LoadIconsKey property to define the starting key when LoadIcons method is used. Use the LoadIcon method to load a single icon to the control's icons list.



property ExFileView.LoadIconsKey as Long

Specifies the starting key when the LoadIcons method is used.

Type	Description
Long	A long expression that indicates the starting key, when LoadIcons method is used.

Use the LoadIconsKey property defines the starting key when the icons will be loaded using the LoadIcons method. Use the LoadIcon method to load a single icon to the control's icons list.

property ExFileView.Loading as String

Specifies the HTML caption being displayed in the list if loading files or folders could take long time.

Type	Description
String	A HTML expression that specifies the caption to be displayed when loading the list of files/folders could take longer.

By default, the Loading property is "Loading...". Use the Loading property to show a caption in the center of the control while loading files/folders. The message shows up only if loading could take longer. The Loading property supports the built-in HTML tags as 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:

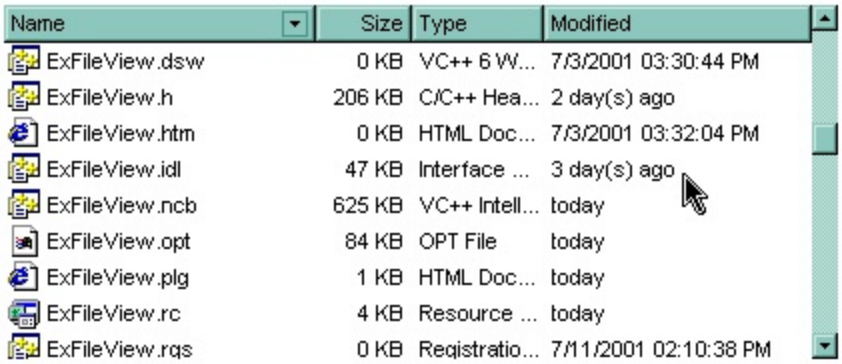
or "*<**sha** 404040;5;0><fgcolor=FFFFFF>outline anti-aliasing</fgcolor> </**sha**>*" gets:

property ExFileView.ModifiedDaysAgo as Long

Specifies a value that indicates whether the Modified column shows the number of days ago when the file was last updated.

Type	Description
Long	A long expression that indicates whether the Modified column shows the number of days ago when the file was last updated.

By default, the ModifiedDaysAgo property is 0. If the ModifiedDaysAgo property is 0, the control displays the date when the file was last updated. If the ModifiedDaysAgo property is greater than zero (**x**), the control displays the message "**x** day(s) ago" (or "today") message for files that were updated in the last x days, else the normal date is displayed. Use the Option property to change the caption for "today" or "x day(s) ago" strings.



method ExFileView.OLEDrag ()

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

Type	Description

This is only for internal use.

property ExFileView.OLEDropMode as exOLEDropModeEnum

Returns or sets how the component handles drop operations.

Type	Description
exOLEDropModeEnum	An exOLEDropModeEnum expression that defines how the component handles the drag and drop operations.

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

property ExFileView.Option(Option as OptionEnum) as Variant

Retrieves or sets a value that indicates an option for the control.

Type	Description
Option as OptionEnum	A long expression that indicates the option of the control being changed.
Variant	A Variant expression that indicates the value for the option being changed. The type of the Variant is based on the Option being changed.

Use the Option property to change a particular option for the control. Please check the OptionEnum enumeration for the options that can be changed. Use the Refresh method to refresh the control's content.

For instance, you can change the format of date being displayed on the 'Modified' column using a VB sample like follows:

```
With ExFileView1
   .Option(exModifiedDateFormat) = "ddd, MMM dd yy "
   .Option(exModifiedTimeFormat) = "HH:mm:ss"
   .Refresh
End With
```

The sample displays the date like: "Thu, Oct 9 04"

The following C++ sample changes the format of date being displayed ib the "Modified" column:

```
m_fileview.SetOption( 2, COleVariant( "ddd, MMM dd yy " ) );
m_fileview.SetOption( 3, COleVariant( "HH:mm:ss" ) );
m_fileview.Refresh();
```

The following VB.NET sample changes the format of date being displayed ib the "Modified" column:

```
With AxExFileView1
   .set_Option(EXFILEVIEWLib.OptionEnum.exModifiedDateFormat, "ddd, MMM dd yy")
   .set_Option(EXFILEVIEWLib.OptionEnum.exModifiedTimeFormat, "HH:mm:ss")
   .CtlRefresh()
End With
```

The following C# sample changes the format of date being displayed ib the "Modified" column:

```
axExFileView1.set_Option(EXFILEVIEWLib.OptionEnum.exModifiedDateFormat, "ddd,
MMM dd yy ");
axExFileView1.set_Option(EXFILEVIEWLib.OptionEnum.exModifiedTimeFormat,
"HH:mm:ss");
axExFileView1.CtlRefresh();
```

The following VFP sample changes the format of date being displayed ib the "Modified" column:

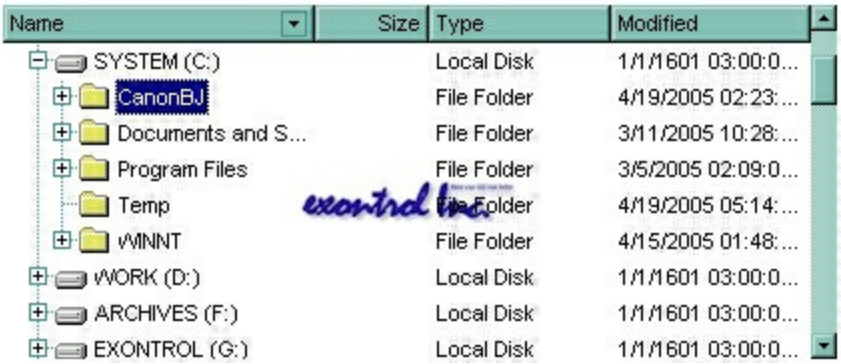
```
With thisform.ExFileView1
   .Option(2) = "ddd, MMM dd yy "   && exModifiedDateFormat
   .Option(3) = "HH:mm:ss"         && exModifiedTimeFormat
   .Object.Refresh
EndWith
```

property ExFileView.Picture as IPictureDisp

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

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

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

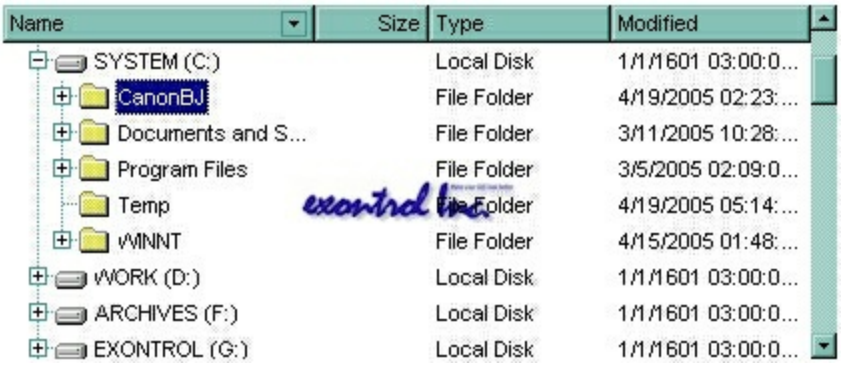


property ExFileView.PictureDisplay as PictureDisplayEnum

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

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

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



method ExFileView.Refresh ()

Refreshes the control.

Type	Description

The Refresh method refreshes the control's content. Use the Apply method to apply rules to the current list.

The following VB sample calls the Refresh method:

```
ExFileView1.Refresh
```

The following C++ sample calls the Refresh method:

```
m_fileview.Refresh();
```

The following VB.NET sample calls the Refresh method:

```
AxExFileView1.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:

```
axExFileView1.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.ExFileView1.Object.Refresh()
```

property ExFileView.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 ExFileView.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 ExFileView.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. The control fires the ScrollButtonClick event when the user clicks a part of the scroll bar.

property ExFileView.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 ExFileView.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.

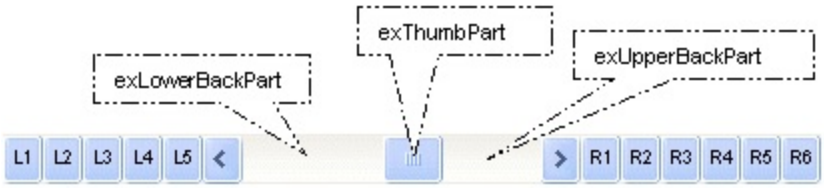
Any other literal between commas is ignored. If duplicate literals are found, the second is ignored, and so on. For instance, "t,l,r" indicates that the left/top and right/bottom buttons are displayed right/bottom after the thumb area.

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

Specifies the caption being displayed on the specified scroll part.

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

Use the ScrolPartCaption property to specify the caption of the scroll's part. Use the ScrollPartVisible property to add or remove buttons/parts in the control's scrollbar. By default, when a part becomes visible, the ScrollPartEnable property is automatically called, so the parts becomes enabled. Use the ScrollPartEnable property to specify enable or disable parts in the control's scrollbar. The control fires the ScrollButtonClick event when the user clicks a part of the scroll bar. Use the ScrollFont property to specify the font in the control's scroll bar. Use the 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 ExFileView1
  .BeginUpdate
```

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

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

```
With AxExFileView1
    .BeginUpdate()
    .set_ScrollPartVisible(EXEXFILEVIEWLib.ScrollBarEnum.exVScroll,
EXEXFILEVIEWLib.ScrollPartEnum.exLeftB1Part Or
EXEXFILEVIEWLib.ScrollPartEnum.exRightB1Part, True)
    .set_ScrollPartCaption(EXEXFILEVIEWLib.ScrollBarEnum.exVScroll,
EXEXFILEVIEWLib.ScrollPartEnum.exLeftB1Part, "<img></img>1")
    .set_ScrollPartCaption(EXEXFILEVIEWLib.ScrollBarEnum.exVScroll,
EXEXFILEVIEWLib.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 :

```
axExFileView1.BeginUpdate();
axExFileView1.set_ScrollPartVisible(EXEXFILEVIEWLib.ScrollBarEnum.exVScroll,
EXEXFILEVIEWLib.ScrollPartEnum.exLeftB1Part |
EXEXFILEVIEWLib.ScrollPartEnum.exRightB1Part, true);
axExFileView1.set_ScrollPartCaption(EXEXFILEVIEWLib.ScrollBarEnum.exVScroll,
EXEXFILEVIEWLib.ScrollPartEnum.exLeftB1Part , "<img></img>1");
axExFileView1.set_ScrollPartCaption(EXEXFILEVIEWLib.ScrollBarEnum.exVScroll,
EXEXFILEVIEWLib.ScrollPartEnum.exRightB1Part, "<img></img>2");
axExFileView1.EndUpdate();
```

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

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

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

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

```
With thisform.ExFileView1
   .BeginUpdate
      .ScrollPartVisible(0, bitor(32768,32)) = .t.
      .ScrollPartCaption(0,32768) = "<img></img>1"
      .ScrollPartCaption(0, 32) = "<img></img>2"
   .EndUpdate
EndWith
```

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

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

property ExFileView.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.

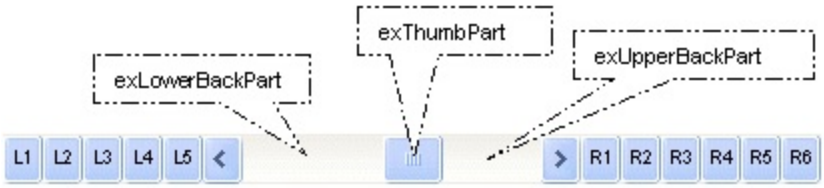
The ScrollPartCaptionAlignment property specifies the alignment of the caption in the part of the scroll bar. By default, the caption is centered. Use the ScrolPartCaption property to specify the caption being displayed on specified part of the scroll bar. Use the ScrollPartVisible property to add or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

property ExFileView.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. The control fires the ScrollButtonClick event when the user clicks a part of the scroll bar. Use the ScrollOrderParts property to customize the order of the buttons in the scroll bar.

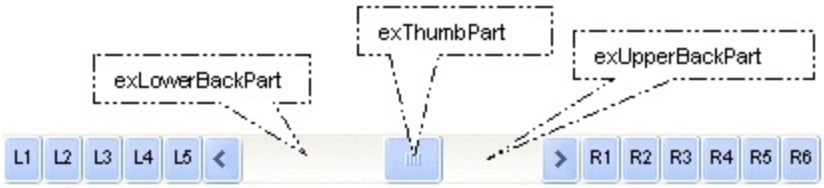


property ExFileView.ScrollPartVisible(ScrollBar as ScrollBarEnum, Part as ScrollPartEnum) as Boolean

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. The control fires the ScrollButtonClick event when the user clicks a part of the scroll bar. 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 ExFileView1
  .BeginUpdate
```

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

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

```
With AxExFileView1
    .BeginUpdate()
    .set_ScrollPartVisible(EXEXFILEVIEWLib.ScrollBarEnum.exVScroll,
EXEXFILEVIEWLib.ScrollPartEnum.exLeftB1Part Or
EXEXFILEVIEWLib.ScrollPartEnum.exRightB1Part, True)
    .set_ScrollPartCaption(EXEXFILEVIEWLib.ScrollBarEnum.exVScroll,
EXEXFILEVIEWLib.ScrollPartEnum.exLeftB1Part, "<img></img>1")
    .set_ScrollPartCaption(EXEXFILEVIEWLib.ScrollBarEnum.exVScroll,
EXEXFILEVIEWLib.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 :

```
axExFileView1.BeginUpdate();
axExFileView1.set_ScrollPartVisible(EXEXFILEVIEWLib.ScrollBarEnum.exVScroll,
EXEXFILEVIEWLib.ScrollPartEnum.exLeftB1Part |
EXEXFILEVIEWLib.ScrollPartEnum.exRightB1Part, true);
axExFileView1.set_ScrollPartCaption(EXEXFILEVIEWLib.ScrollBarEnum.exVScroll,
EXEXFILEVIEWLib.ScrollPartEnum.exLeftB1Part , "<img></img>1");
axExFileView1.set_ScrollPartCaption(EXEXFILEVIEWLib.ScrollBarEnum.exVScroll,
EXEXFILEVIEWLib.ScrollPartEnum.exRightB1Part, "<img></img>2");
axExFileView1.EndUpdate();
```

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

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

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

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

```
With thisform.ExFileView1
   .BeginUpdate
      .ScrollPartVisible(0, bitor(32768,32)) = .t.
      .ScrollPartCaption(0,32768) = "<img></img>1"
      .ScrollPartCaption(0, 32) = "<img></img>2"
   .EndUpdate
EndWith
```

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

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

property ExFileView.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 ExFileView.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 ExFileView1_OffsetChanged(ByVal Horizontal As Boolean, ByVal NewVal As
Long)
   If (Not Horizontal) Then
      ExFileView1.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 AxExFileView1_OffsetChanged(ByVal sender As System.Object, ByVal e As
AxEXEXFILEVIEWLib._IExFileViewEvents_OffsetChangedEvent) Handles
AxExFileView1.OffsetChanged
   If (Not e.horizontal) Then
      AxExFileView1.set_ScrollToolTip(EXEXFILEVIEWLib.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 OnOffsetChangedExFileView1(BOOL Horizontal, long NewVal)
{
   if ( !Horizontal )
   {
      CString strFormat;
      strFormat.Format( _T("%i"), NewVal );
      m_fileView.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 axExFileView1_OffsetChanged(object sender,
AxEXEXFILEVIEWLib._IExFileViewEvents_OffsetChangedEvent e)
{
   if ( !e.horizontal )
      axExFileView1.set_ScrollToolTip(EXEXFILEVIEWLib.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.ExFileView1.ScrollToolTip(0) = "Record " + ltrim(str(newval))
EndIf
```

property ExFileView.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 ExFileView.Search as String

Specifies the list of files and folders including wild card characters to search for.

Type	Description
String	A string expression that indicates the list of files or folders to search for.

Use the Search property to get the list of files and folders that match a pattern or multiple patterns. The Search property includes the patterns separated them by space character. A pattern may include wild characters like : '*' (Zero or more characters) or '?' (Any single character). For instance, the '*.txt *.doc' specifies all files with extension 'txt' and files with 'doc' extension. Use the Add method to add rules to customize the found items. Use the StopSearch method to stop immediately searching the files. The Search event is fired when searching files starts or when searching the files ends. The Search = "" stops searching files and restores the control's content to browse the ExploreFromHere path. Use the Get method to get the list of files and folders in the control.

The following VB sample gets the list of document files and text files:

```
Private Sub Command1_Click()
   ExFileView1.Search = "*.doc *.txt"
End Sub

Private Sub ExFileView1_Search(ByVal State As EXFILEVIEWLibCtl.SearchStateEnum)
   Select Case State
      Case 0
         Debug.Print "Searching for '" & ExFileView1.Search & "' starts."
      Case 1
         Debug.Print "Searching for '" & ExFileView1.Search & "' ends."
   End Select
End Sub
```

The following VB sample searches for the '.cpp' files and get them in red color, and '.h' files and get them in blue color:

```
Private Sub Command1_Click()
   With ExFileView1
      With .FileTypes
         With .Add("*.cpp")
```

```
            .ForeColor = vbRed
        End With
        With .Add("*.h")
            .ForeColor = vbBlue
        End With
    End With
    .Search = "*.cpp *.h"
  End With
End Sub

Private Sub ExFileView1_Search(ByVal State As EXFILEVIEWLibCtl.SearchStateEnum)
  Select Case State
    Case 0
        Debug.Print "Searching for '" & ExFileView1.Search & "' starts."
    Case 1
        Debug.Print "Searching for '" & ExFileView1.Search & "' ends."
  End Select
End Sub
```

property ExFileView.SelBackColor as Color

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

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

The SelBackColor property specifies the background color for selected items. Use the SelForeColor property to specify the foreground color for selected items. Use the Get method to access the selected items collection. Use the BackColor property to specify the control's background color. Use the ForeColor property to specify the control's foreground color. Use the BackColor and ForeColor properties to change the background and foreground colors for a specified file/folder. Use the Selected property to specify whether a File object is selected or unselected. How do I assign a new look for the selected item?

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

```
With ExFileView1
   With .VisualAppearance
      .Add &H23, App.Path + "\selected.ebn"
   End With
   .SelForeColor = RGB(0, 0, 0)
   .SelBackColor = &H23000000
End With
```

The sample adds the skin with the index 35 (Hexa 23), and applies to the selected item using the SelBackColor property.

The following C++ sample applies a new appearance to the selected item(s):

```
#include "Appearance.h"
```

```
m_fileview.GetVisualAppearance().Add( 0x23,
COleVariant(_T("D:\\Temp\\ExFileView_Help\\selected.ebn")) );
m_fileview.SetSelBackColor( 0x23000000 );
m_fileview.SetSelForeColor( 0 );
```

The following VB.NET sample applies a new appearance to the selected item(s):

```
With AxExFileView1
   With .VisualAppearance
      .Add(&H23, "D:\Temp\ExFileView_Help\selected.ebn")
   End With
   .SelForeColor = Color.Black
   .Template = "SelBackColor = 587202560"
End With
```

The VB.NET sample uses the Template property to assign a new value to the SelBackColor property. The 587202560 value represents &23000000 in hexadecimal.

The following C# sample applies a new appearance to the selected item(s):

```
axExFileView1.VisualAppearance.Add(0x23, "D:\\Temp\\ExFileView_Help\\selected.ebn");
axExFileView1.Template = "SelBackColor = 587202560";
```

The following VFP sample applies a new appearance to the selected item(s):

```
With thisform.ExFileView1
   With .VisualAppearance
      .Add(35, "D:\Temp\ExFileView_Help\selected.ebn")
   EndWith
   .SelForeColor = RGB(0, 0, 0)
   .SelBackColor = 587202560
EndWith
```

The 587202560 value represents &23000000 in hexadecimal. The 32 value represents &23 in hexadecimal

How do I assign a new look for the selected item?

The component supports skinning parts of the control, including the selected item. Shortly, the idea is that identifier of the skin being added to the Appearance collection is stored in

the first significant byte of property of the color type. In our case, we know that the SelBackColor property changes the background color for the selected item. This is what we need to change. In other words, we need to change the visual appearance for the selected item, and that means changing the background color of the selected item. So, the following code (blue code) changes the appearance for the selected item:

```
With ExFileVw1
   .VisualAppearance.Add &H34, App.Path + "\aqua.ebn"
   .SelBackColor = &H34000000
End With
```

Please notice that the 34 hexa value is arbitrary chosen, it is not a predefined value. Shortly, we have added a skin with the identifier 34, and we specified that the SelBackColor property should use that skin, in order to change the visual appearance for the selected item. Also, please notice that the 34 value is stored in the first significant byte, not in other position. For instance, the following sample doesn't use any skin when displaying the selected item:

```
With ExFileVw1
   .VisualAppearance.Add &H34, App.Path + "\aqua.ebn"
   .SelBackColor = &H34
End With
```

This code (red code) DOESN'T use any skin, because the 34 value is not stored in the higher byte of the color value. The sample just changes the background color for the selected item to some black color (RGB(0,0,34)). So, please pay attention when you want to use a skin and when to use a color. Simple, if you are calling &H**34**000000, you have 34 followed by 6 (six) zeros, and that means the first significant byte of the color expression. Now, back to the problem. The next step is how we are creating skins? or EBN files? The Exontrol's exbutton component includes a builder tool that saves skins to EBN files. So, if you want to create new skin files, you need to download and install the exbutton component from our web site. Once that the exbutton component is installed, please follow the steps.

Let's say that we have a BMP file, that we want to stretch on the selected item's background.

1. Open the VB\Builder or VC\Builder sample
2. Click the **New File** button (on the left side in the toolbar), an empty skin is created.
3. Locate the **Background** tool window and select the **Picture\Add New** item in the menu, the Open file dialog is opened.
4. Select the picture file (GIF, BMP, JPG, JPEG). You will notice that the visual

appearance of the focused object in the skin is changed, actually the picture you have selected is tiled on the object's background.

5. Select the **None** item, in the Background tool window, so the focused object in the skin is not displaying anymore the picture being added.
6. Select the **Root** item in the skin builder window (in the left side you can find the hierarchy of the objects that composes the skin), so the Root item is selected, and so focused.
7. Select the picture file you have added at the step 4, so the Root object is filled with the picture you have chosen.
8. Resize the picture in the Background tool window, until you reach the view you want to have, no black area, or change the CX and CY fields in the Background tool window, so no black area is displayed.
9. Select **Stretch** button in the Background tool window, so the Root object stretches the picture you have selected.
10. Click the **Save a file** button, and select a name for the new skin, click the Save button after you typed the name of the skin file. Add the .ebn extension.
11. Close the builder

You can always open the skin with the builder and change it later, in case you want to change it.

Now, create a new project, and insert the component where you want to use the skin, and add the skin file to the Appearance collection of the object, using blue code, by changing the name of the file or the path where you have selected the skin. Once that you have added the skin file to the Appearance collection, you can change the visual appearance for parts of the controls that supports skinning. **Usually the properties that changes the background color for a part of the control supports skinning as well.**

method ExFileView.Select (Folder as String)

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

Type	Description
Folder as String	A String expression that specifies the folder to be selected.

The Select method selects a folder, giving its displaying name, relative or absolute path.

property ExFileView.SelectByDrag as Boolean

Specifies whether the user selects multiple items by dragging.

Type	Description
Boolean	A boolean expression that specifies whether the user may select multiple items by drag and drop.

By default, SelectByDrag property is True. Use the SelectByDrag property to disable selecting multiple items by dragging. The SelectByDrag property has effect only if the control supports multiple selection. The SingleSel property controls the number of items that the user may select. For instance, if the SingleSel property is True, the user can't select multiple items, and so a single item may be selected at the time. If the SingleSel property is False, the user can select multiple items using the mouse, keyboard or both. When the SelectByDrag property is True, the user may click the non text area to start select items by dragging. Use the SelectByDrag property on False when your control requires OLE drag and drop operations, like when you select multiple items and drag them to a new position. Use the OLEDropMode property to specify whether the OLE drag and drop operations inside the control is allowed. For instance, if the SelectByDrag and OLEDropMode properties are on, sometimes it is confused what control should do when user clicks and start to select items. The AllowSelectNothing property specifies whether the current selection is erased, once the user clicks outside of the items section. The SelectOnRelease property indicates whether the selection occurs when the user releases the mouse button.

property ExFileView.SelectOnRelease as Boolean

Indicates whether the selection occurs when the user releases the mouse button.

Type	Description
Boolean	A Boolean expression that indicates whether the selection occurs when the user releases the mouse button.

By default, the SelectOnRelease property is False. By default, the selection occurs, as soon as the user clicks an object. The SelectOnRelease property indicates whether the selection occurs when the user releases the mouse button. The SelectOnRelease property has no effect if the SingleSel property is False, and SelectByDrag property is True.

property ExFileView.SelForeColor as Color

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

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

The SelForeColor property specifies the foreground color for selected items. Use the SelBackColor property to specify the background color for selected items. Use the Get method to access the selected items collection. Use the ForeColor property to specify the control's foreground color. Use the BackColor property to specify the control's background color. Use the Selected property to specify whether a File object is selected or unselected.

property ExFileView.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.

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-

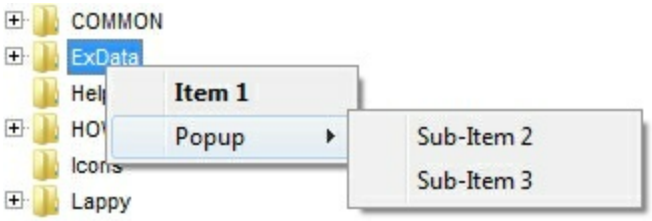
selected item in the control (the keyword's value is valid while the filecount property is 1)

- **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 ExFileView.ShowFocusRect as Boolean

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

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

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

The following screen shot shows the one focused item (ExData), and 2 selected items (COMMON and HOW-TO-Before):

property ExFileView.SingleSel as Boolean

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

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

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

The following VB enumerates the selected files and folders:

```
Dim i As Long
With ExFileView1.Get(SelItems)
   For i = 0 To .Count - 1
      Debug.Print .Item(i).FullName
   Next
End With
```

The following C++ enumerates the selected files and folders:

```
CFiles files = m_fileview.GetGet( 0 /*SelItems*/ );
for ( long i = 0; i < files.GetCount(); i++ )
   OutputDebugString( files.GetItem( COleVariant( i ) ).GetFullName() );
```

The following VB.NET enumerates the selected files and folders:

```
With AxExFileView1.get_Get(EXFILEVIEWLib.TypeEnum.SelItems)
   Dim i As Integer
   For i = 0 To .Count - 1
      Debug.WriteLine(.Item(i).FullName)
   Next
End With
```

The following C# enumerates the selected files and folders:

```
EXFILEVIEWLib.Files files = axExFileView1.get_Get(EXFILEVIEWLib.TypeEnum.SelItems);
for (int i = 0; i < files.Count; i++)
{
    EXFILEVIEWLib.File file = files[i];
    System.Diagnostics.Debug.WriteLine(file.FullName);
}
```

The following VFP enumerates the selected files and folders:

```
with thisform.ExFileView1.Get( 0 ) && SelItems
    local i
    for i = 0 to .Count - 1
        with .Item(i)
            wait window nowait .FullName
        endwith
    next
 endwith
```

method ExFileView.Sort (ColumnName as String, Ascending as Boolean)

Sorts a column.

Type	Description
ColumnName as String	A string expression that indicates the column being sorted. The valid values are: 'Name', 'Size', 'Modified', 'Type', 'In Folder'.
Ascending as Boolean	A boolean expression that indicates how the column is sorted. True means ascending, False means descending.

Use the Sort method to sort the column at runtime. Use the HeaderVisible property to specify whether the control's header bar is visible or hidden. The user can select a column by clicking the column's caption. Use the ColumnVisible property to hide or show a column. Use the ColumnCaption property to specify the column's caption. Use the ColumnWidth property to change the column's width. Use the BackColorHeader property to specify the background color for the control's header bar. Use the ForeColorHeader property to specify the foreground color for the control's header bar.

property ExFileView.Statistics as String

Gives statistics data of objects being hold by the control.

Type	Description
String	A String expression that gives statistics data of objects being hold by the control.

The Statistics property gives statistics information of objects being hold by the control.

method ExFileView.StopSearch ()

Stops the searching operation.

Type	Description

Stops the searching files. Use the Search property to search for files or folders that matches a pattern or multiple patterns. The Search event is fired when searching files starts or when searching the files ends. Use the Get method to get the list of files and folders in the control.

The following VB sample displays the list of files as they are displayed:

```
With ExFileView1.Get(VisibleItems)
   For i = 0 To .Count - 1
      With .Item(i)
         Debug.Print .Name
      End With
   Next
End With
```

The following C++ sample displays the list of files as they are displayed:

```
CFiles files = m_fileview.GetGet( 3 /*VisibleItems*/ );
for ( long i = 0; i < files.GetCount(); i++ )
   OutputDebugString( files.GetItem( COleVariant( i ) ).GetName() );
```

The following VB.NET sample displays the list of files as they are displayed:

```
With AxExFileView1.get_Get(EXFILEVIEWLib.TypeEnum.VisibleItems)
   Dim i As Integer
   For i = 0 To .Count - 1
      With .Item(i)
         Debug.WriteLine(.Name())
      End With
   Next
End With
```

The following C# sample displays the list of files as they are displayed:

```
EXFILEVIEWLib.Files files = axExFileView1.get_Get(EXFILEVIEWLib.TypeEnum.VisibleItems);
```

```
for (int i = 0; i < files.Count; i++)
{
    EXFILEVIEWLib.File file = files[i];
    System.Diagnostics.Debug.WriteLine(file.Name);
}
```

The following VFP sample displays the list of files as they are displayed:

```
With thisform.ExFileView1.Get(3)    && VisibleItems
    For i = 0 To .Count - 1
        With .Item(i)
            wait window nowait .Name
        EndWith
    Next
EndWith
```

property ExFileView.Template as String

Specifies the control's template.

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

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

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

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

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

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

An instruction can be one of the following:

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

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

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

The Template supports the following general functions:

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

property ExFileView.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 ExFileView.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:

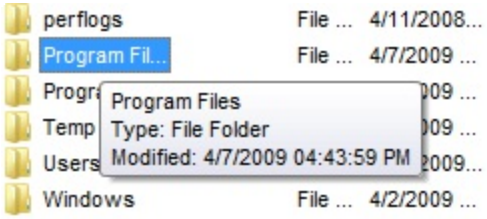
- **Me** *property indicates the original object.*
- **RGB(**R,G,B**)** *property retrieves an RGB value, where the R, G, B are byte values that indicates the R G B values for the color being specified. For instance, the following code changes the control's background color to red: BackColor = RGB(255,0,0)*
- **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 ExFileView.ToolTipDelay as Long

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

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

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

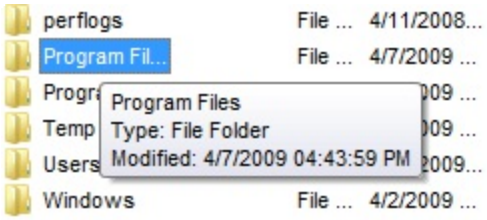


property ExFileView.ToolTipFont as IFontDisp

Retrieves or sets the tooltip's font.

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

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

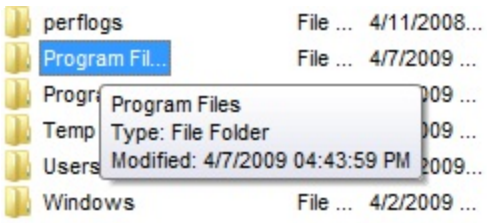


property ExFileView.ToolTipPopDelay as Long

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

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

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

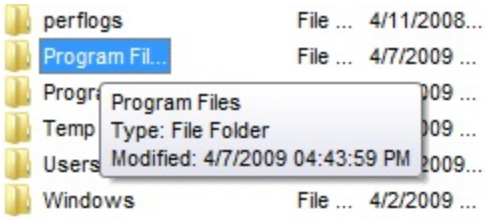


property ExFileView.ToolTipWidth as Long

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

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

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



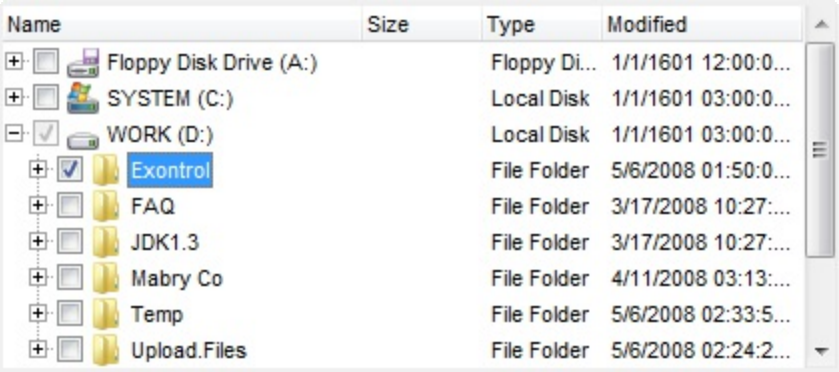
property ExFileView.UseVisualTheme as UIVisualThemeEnum

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

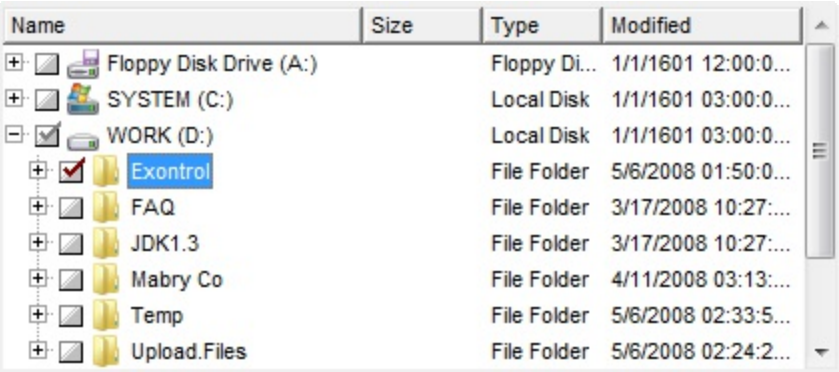
Type	Description
UIVisualThemeEnum	An UIVisualThemeEnum expression that specifies which UI parts of the control are shown using the current visual theme.

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

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



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



property ExFileView.Version as String

Retrieves the control's version.

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

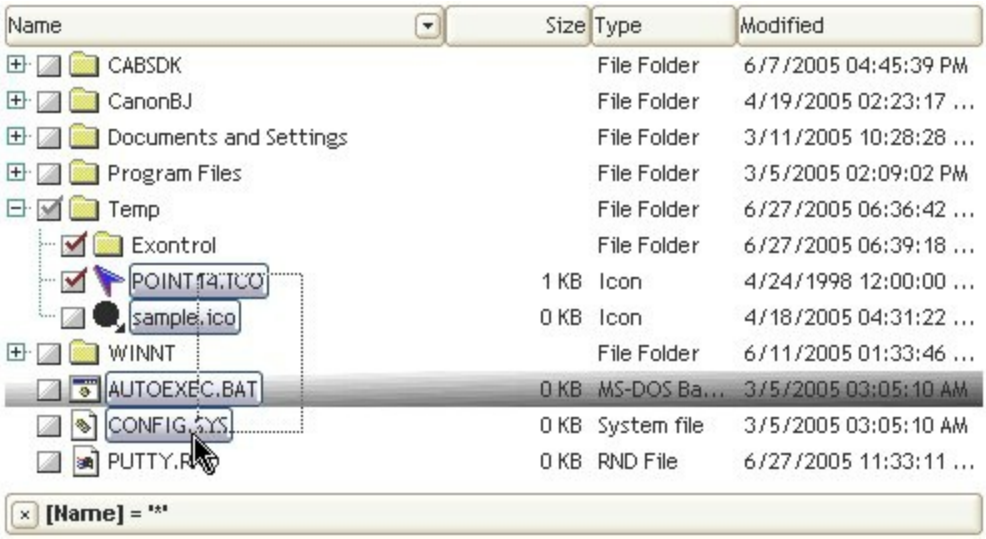
The Version property is read-only. The Version property specifies the version of the control you are running.

property ExFileView.VisualAppearance as Appearance

Retrieves the control's appearance. /*not supported in the lite version*/

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

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



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

- selected file/folder, SelBackColor property
- control's **header bar**, BackColorHeader property
- control's **filter bar**, FilterBarBackColor property
- "**drop down**" filter bar button, "close" filter bar button, and so on, Background property

File object

A File object stores information about a file or a folder. To access a File object you have to use the Get property, that retrieves a Files collection. The Files object contains a collection of File objects. For instance the following sample shows how to select by code the "WinNT" folder: ExFileView1.Get(AllItems)("WinNT").Selected = True. The File object supports the following properties and methods.

Name	Description
Accessed	Retrieves the date when the file was last read from, written to, or for executable files, run.
BackColor	Returns or sets the background color of the file.
Bold	Specifies whether the file should appear in bold.
Checked	Specifies whether the file is checked or unchecked.
Children	Returns a collection with child objects of the current object.
Created	Retrieves the date when the file or directory was created.
Folder	Specifies whether the object is a file or a folder.
ForeColor	Returns or sets the foreground color of the file.
FullName	Retrieves the full name of the file.
Ghosted	Returns or sets a value that determines whether a file appears as unavailable (it appears dimmed).
Modified	Retrieves the date when the file was last written to, truncated, or overwritten.
Name	Retrieves the file's name.
ParseName	Retrieves the file's parse name.
RelativeName	Retrieves the relative name of the file.
Selected	Specifies whether the file is selected or unselected.
Size	Retrieves the size of the file.
State	Indicates the file's changed state.
Type	Retrieves or sets a value that indicates the string gets displayed on the Type column.

property File.Accessed as Date

Retrieves the date when the file was last read from, written to, or for executable files, run.

Type	Description
Date	A DATE expression that indicates the date when the file was last read from, written to, or for executable files, run.

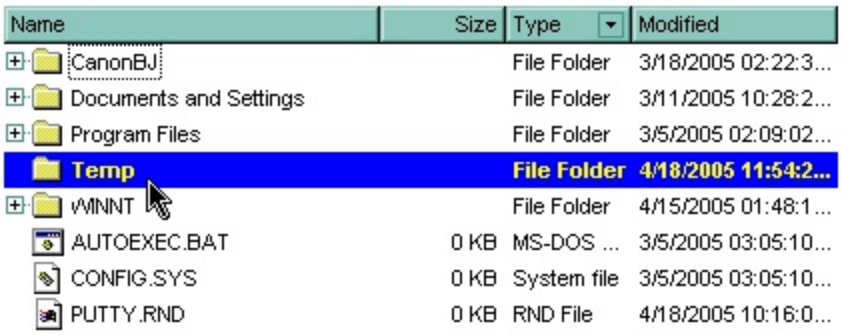
The Accessed property specifies the date when the file was last read from, written to, or for executable files, run. The Size property gets the size of the file in bytes. The Size property gets 0, for folder objects. The FullName property indicates the fully name of the file or directory. The Modified property specifies the date when the file was last written to, truncated, or overwritten. The Created property specifies the date when the file or directory was created.

property File.BackColor as Color

Returns or sets the background color of the file.

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

Use the BackColor and ForeColor properties to change the background and foreground colors for a specified file/folder. Use the Get method to get the collection of files/folders. Use the Folder property to specify whether the File object hosts a file or a folder. Use the SelBackColor property to specify the background color for selected items. Use the Ghosted property to change the item's appearance.



The following VB sample changes the background color for the item named "Temp":

```
With ExFileView1.Get(AllItems)("Temp")
   .BackColor = vbBlue
   .ForeColor = vbYellow
   .Bold = True
End With
```

The BackColor attribute is lost if the control is refreshed or if the Apply property is called. For instance, If you need a permanent background color for specified folders, you can add a new FileType object like in the following VB sample:

```
With ExFileView1.FileTypes.Add("Temp")
   .Folder = True
   .BackColor = vbBlue
   .ForeColor = vbYellow
   .Bold = True
   .Apply
End With
```

The following C++ sample changes the background color for the item named "Temp":

```
CFile1 file = m_fileview.GetGet( 1 /*AllItems*/ ).GetItem( COleVariant( "Temp" ) );
file.SetBackColor( RGB(0,0,255) );
file.SetForeColor( RGB(255,255,255) );
file.SetBold( TRUE );
```

The BackColor attribute is lost if the control is refreshed or if the Apply property is called. For instance, If you need a permanent background color for specified folders, you can add a new FileType object like in the following C++ sample:

```
#include "FileType.h"
#include "FileTypes.h"
CFileType fileType = m_fileview.GetFileTypes().Add("Temp");
fileType.SetFolder( TRUE );
fileType.SetBold( TRUE );
fileType.SetForeColor( RGB(255,255,255) );
fileType.SetBackColor( RGB(0,0,255) );
fileType.Apply();
```

The following VB.NET sample changes the background color for the item named "Temp":

```
With AxExFileView1.get_Get(EXFILEVIEWLib.TypeEnum.AllItems).Item("Temp")
   .BackColor = ToUInt32(Color.Blue)
   .ForeColor = ToUInt32(Color.Yellow)
   .Bold = True
End With
```

where the ToUInt32 function converts a Color expression to OLE_COLOR,

```
Shared Function ToUInt32(ByVal c As Color) As UInt32
   Dim i As Long
   i = c.R
   i = i + 256 * c.G
   i = i + 256 * 256 * c.B
   ToUInt32 = Convert.ToUInt32(i)
End Function
```

The BackColor attribute is lost if the control is refreshed or if the Apply property is called. For instance, If you need a permanent background color for specified folders, you can add

a new FileType object like in the following VB.NET sample:

```
With AxExFileView1.FileTypes.Add("Temp")
    .Folder = True
    .BackColor = ToUInt32(Color.Blue)
    .ForeColor = ToUInt32(Color.Yellow)
    .Bold = True
    .Apply()
End With
```

The following C# sample changes the background color for the item named "Temp":

```
EXFILEVIEWLib.File file = axExFileView1.get_Get(EXFILEVIEWLib.TypeEnum.AllItems)
["Temp"];
file.BackColor = ToUInt32(Color.Blue);
file.ForeColor = ToUInt32(Color.Yellow);
file.Bold = true;
```

where the ToUInt32 function converts a Color expression to OLE_COLOR,

```
private UInt32 ToUInt32(Color c)
{
    long i;
    i = c.R;
    i = i + 256 * c.G;
    i = i + 256 * 256 * c.B;
    return Convert.ToUInt32(i);
}
```

The BackColor attribute is lost if the control is refreshed or if the Apply property is called. For instance, If you need a permanent background color for specified folders, you can add a new FileType object like in the following C# sample:

```
EXFILEVIEWLib.FileType fileType = axExFileView1.FileTypes.Add("Temp");
fileType.Folder = true;
fileType.BackColor = ToUInt32(Color.Blue);
fileType.ForeColor = ToUInt32(Color.Yellow);
fileType.Bold = true;
fileType.Apply();
```

The following VFP sample changes the background color for the item named "Temp":

```
With thisform.ExFileView1.Get(1).Item("Temp") && AllItems
   .BackColor = RGB(0,0,255)
   .ForeColor = RGB(255,255,255)
   .Bold = .t.
EndWith
```

The BackColor attribute is lost if the control is refreshed or if the Apply property is called. For instance, If you need a permanent background color for specified folders, you can add a new FileType object like in the following VFP sample:

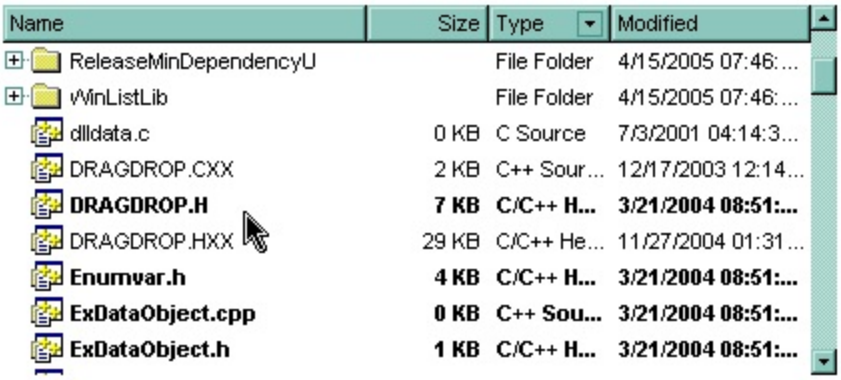
```
With thisform.ExFileView1.FileTypes.Add("Temp")
   .Folder = .t.
   .BackColor = RGB(0,0,255)
   .ForeColor = RGB(255,255,255)
   .Bold = .t.
   .Apply()
EndWith
```

property File.Bold as Boolean

Specifies whether the file should appear in bold.

Type	Description
Boolean	A boolean expression that indicates whether the file should appear in bold.

Use the Bold property to bold files and folders. Use the Font property to specify the control's font. Use the Get property to retrieve the items collection. Use the BrowseFolderPath property to specify the path to the browsed folder.



The following VB sample bolds the cpp and h files in the browsed folder:

```
Dim fs As Files, f As File
Set fs = ExFileView1.Get(AllItems).Get("*.cpp *.h")
For Each f In fs
   f.Bold = True
Next
```

The Bold attribute is lost if the control is refreshed, or if the Apply property is invoked. For instance, you can add a new FileType object that bolds the cpp and h files:

```
With ExFileView1.FileTypes.Add("*.cpp *.h")
   .Bold = True
   .Apply
End With
```

The following C++ sample bolds the cpp and h files in the browsed folder:

```
CFiles files = m_fileview.GetGet( 1 /*AllItems*/ ).GetGet( "*.cpp *.h" );
for ( long i = 0; i < files.GetCount(); i++ )
```

```
files.GetItem( COleVariant( i ) ).SetBold( TRUE );
```

The Bold attribute is lost if the control is refreshed, or if the Apply property is invoked. For instance, you can add a new FileType object that bolds the cpp and h files:

```
#include "FileType.h"
#include "FileTypes.h"
CFileType fileType = m_fileview.GetFileTypes().Add("*.cpp *.h");
fileType.SetBold( TRUE );
fileType.Apply();
```

The following VB.NET sample bolds the cpp and h files in the browsed folder:

```
With AxExFileView1.get_Get(EXFILEVIEWLib.TypeEnum.AllItems).Get("*.cpp *.h")
   Dim i As Integer
   For i = 0 To .Count - 1
      .Item(i).Bold = True
   Next
End With
```

The Bold attribute is lost if the control is refreshed, or if the Apply property is invoked. For instance, you can add a new FileType object that bolds the cpp and h files:

```
With AxExFileView1.FileTypes.Add("*.cpp *.h")
   .Bold = True
   .Apply()
End With
```

The following C# sample bolds the cpp and h files in the browsed folder:

```
EXFILEVIEWLib.Files files =
axExFileView1.get_Get(EXFILEVIEWLib.TypeEnum.AllItems).get_Get("*.cpp *.h");
for (int i = 0; i < files.Count; i++)
   files[i].Bold = true;
```

The Bold attribute is lost if the control is refreshed, or if the Apply property is invoked. For instance, you can add a new FileType object that bolds the cpp and h files:

```
EXFILEVIEWLib.FileType fileType = axExFileView1.FileTypes.Add("*.cpp *.h");
fileType.Bold = true;
```

```
fileType.Apply();
```

The following VFP sample bolds the cpp and h files in the browsed folder:

```
With thisform.ExFileView1.Get( 1 ).Get("*.cpp *.h")
   local i
   for i = 0 to .Count - 1
      with .Item(i)
         .Bold = .t.
      endwith
   next
EndWith
```

The Bold attribute is lost if the control is refreshed, or if the Apply property is invoked. For instance, you can add a new FileType object that bolds the cpp and h files:

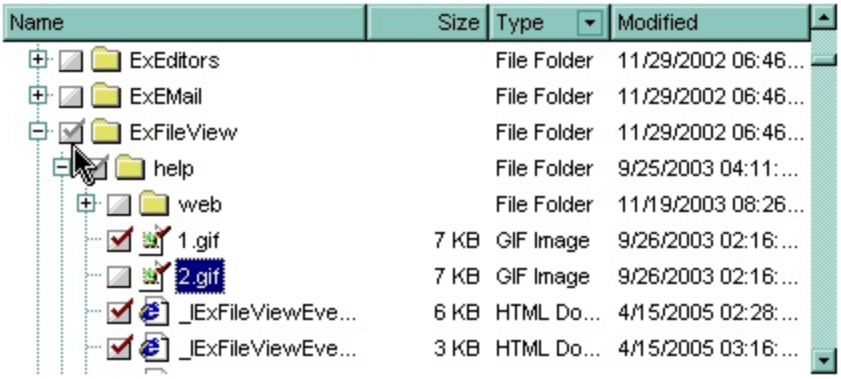
```
With thisform.ExFileView1.FileTypes.Add("*.cpp *.h")
   .Bold = .t.
   .Apply()
EndWith
```

property File.Checked as Boolean

Specifies whether the file is checked or unchecked.

Type	Description
Boolean	A boolean expression that indicates whether a file or folder is checked or unchecked.

Use the Checked property to determine whether a file is checked or unchecked. Use the HasCheckBox property to assign a check box for each item in your control. Use the Get method to get the collection of checked items like shown in the following samples. Use the Folder property to specify whether a File object holds a file or a folder. Use the Name property to get the name of the file or folder. Use the BrowseFolderPath property to specify the path to the browsed folder.



The following VB sample displays the checked files and folders:

```
With ExFileView1.Get(CheckItems)
   Dim i As Long
   For i = 0 To .Count - 1
      With .Item(i)
         Debug.Print .Name
      End With
   Next
End With
```

The following VB sample checks the selected file, by code:

```
ExFileView1.Get(SelItems)(0).Checked = True
```

The following C++ sample displays the checked files and folders:

```
CFiles files = m_fileview.GetGet( 2 /*CheckItems*/ );
```

```
for ( long i = 0; i < files.GetCount(); i++ )
   OutputDebugString( files.GetItem( COleVariant( i ) ).GetName() );
```

The following VB.NET sample displays the checked files and folders:

```
With AxExFileView1.get_Get(EXFILEVIEWLib.TypeEnum.CheckItems)
   Dim i As Integer
   For i = 0 To .Count - 1
      Debug.WriteLine(.Item(i).Name)
   Next
End With
```

The following C# sample displays the checked files and folders:

```
EXFILEVIEWLib.Files files = axExFileView1.get_Get(EXFILEVIEWLib.TypeEnum.CheckItems);
for (int i = 0; i < files.Count; i++)
   System.Diagnostics.Debug.WriteLine( files[i].Name );
```

The following VFP sample displays the checked files and folders:

```
With thisform.ExFileView1.Get( 2 ) && CheckItems
   local i
   for i = 0 to .Count - 1
      with .Item(i)
         wait window nowait .Name
      endwith
   next
EndWith
```

property File.Children as Files

Returns a collection with child objects of the current object.

Type	Description
Files	A Files collection that includes the files and sub-folders of the current folder object.

The Children property retrieves an empty collection, if the current object is a file. The Folder property specifies whether the current object is a folder or a file. A Folder object may returns empty collection if no files or sub-folders are included. You can use the Children property to enumerate recursively the files and folders.

The Files collection supports for each statement, so for instance, the following sample displays the list of files/folders of the current selection:

```
Private Sub ExFileView1_StateChange(ByVal State As EXFILEVIEWLibCtl.StateChangeEnum)
  If (State = SelChangeState) Then
    With ExFileView1.Get(SelItems)
      If (.Count > 0) Then
        With .Item(0)
          If (.Folder) Then
            Dim f As EXFILEVIEWLibCtl.File
            For Each f In .Children
              Debug.Print "Sub-Files/Folders: " & f.Name
            Next
          End If
        End With
      End If
    End With
  End If
End Sub
```

property File.Created as Date

Retrieves the date when the file or directory was created.

Type	Description
Date	A DATE expression that specifies the date when the file or directory was created.

The Created property specifies the date when the file or directory was created. The Size property gets the size of the file in bytes. The FullName property indicates the fully name of the file or directory. The Modified property specifies the date when the file was last written to, truncated, or overwritten. The Accessed property specifies the date when the file was last read from, written to, or for executable files, run.

property File.Folder as Boolean

Specifies whether the object is a file or a folder.

Type	Description
Boolean	A boolean expression that specifies whether the object contains a file or a folder. True, the object is a folder, False, the object is a file.

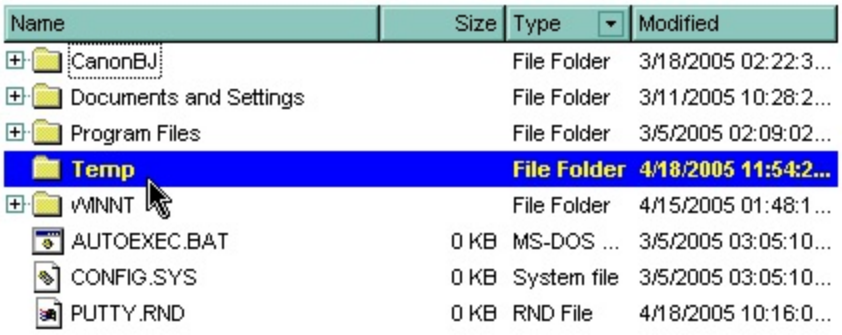
The Folder property is read-only. Use the Folder property to check the type of object stored by the File object. Use the Name property to specify the name of the file or folder. Use the FullName property to get the full name of the file or folder. Use the Folder property to specify whether the select only folders.

property File.ForeColor as Color

Returns or sets the foreground color of the file.

Type	Description
Color	A color expression that defines the file's foreground color.

Use the BackColor and ForeColor properties to change the background and foreground colors for a specified file/folder. Use the Get method to get the collection of files/folders. Use the Folder property to specify whether the File object hosts a file or a folder. Use the SelForeColor property to specify the foreground color for selected items.



The following VB sample changes the foreground color for the item named "Temp":

```
With ExFileView1.Get(AllItems)("Temp")
   .BackColor = vbBlue
   .ForeColor = vbYellow
   .Bold = True
End With
```

The BackColor attribute is lost if the control is refreshed or if the Apply property is called. For instance, If you need a permanent foreground color for specified folders, you can add a new FileType object like in the following VB sample:

```
With ExFileView1.FileTypes.Add("Temp")
   .Folder = True
   .BackColor = vbBlue
   .ForeColor = vbYellow
   .Bold = True
   .Apply
End With
```

The following C++ sample changes the foreground color for the item named "Temp":

```
CFile1 file = m_fileview.GetGet( 1 /*AllItems*/ ).GetItem( COleVariant( "Temp" ) );
file.SetBackColor( RGB(0,0,255) );
file.SetForeColor( RGB(255,255,255) );
file.SetBold( TRUE );
```

The BackColor attribute is lost if the control is refreshed or if the Apply property is called. For instance, If you need a permanent foreground color for specified folders, you can add a new FileType object like in the following C++ sample:

```
#include "FileType.h"
#include "FileTypes.h"
CFileType fileType = m_fileview.GetFileTypes().Add("Temp");
fileType.SetFolder( TRUE );
fileType.SetBold( TRUE );
fileType.SetForeColor( RGB(255,255,255) );
fileType.SetBackColor( RGB(0,0,255) );
fileType.Apply();
```

The following VB.NET sample changes the foreground color for the item named "Temp":

```
With AxExFileView1.get_Get(EXFILEVIEWLib.TypeEnum.AllItems).Item("Temp")
   .BackColor = ToUInt32(Color.Blue)
   .ForeColor = ToUInt32(Color.Yellow)
   .Bold = True
End With
```

where the ToUInt32 function converts a Color expression to OLE_COLOR,

```
Shared Function ToUInt32(ByVal c As Color) As UInt32
   Dim i As Long
   i = c.R
   i = i + 256 * c.G
   i = i + 256 * 256 * c.B
   ToUInt32 = Convert.ToUInt32(i)
End Function
```

The BackColor attribute is lost if the control is refreshed or if the Apply property is called. For instance, If you need a permanent foreground color for specified folders, you can add a new FileType object like in the following VB.NET sample:

```
With AxExFileView1.FileTypes.Add("Temp")
    .Folder = True
    .BackColor = ToUInt32(Color.Blue)
    .ForeColor = ToUInt32(Color.Yellow)
    .Bold = True
    .Apply()
End With
```

The following C# sample changes the foreground color for the item named "Temp":

```
EXFILEVIEWLib.File file = axExFileView1.get_Get(EXFILEVIEWLib.TypeEnum.AllItems)
["Temp"];
file.BackColor = ToUInt32(Color.Blue);
file.ForeColor = ToUInt32(Color.Yellow);
file.Bold = true;
```

where the ToUInt32 function converts a Color expression to OLE_COLOR,

```
private UInt32 ToUInt32(Color c)
{
    long i;
    i = c.R;
    i = i + 256 * c.G;
    i = i + 256 * 256 * c.B;
    return Convert.ToUInt32(i);
}
```

The BackColor attribute is lost if the control is refreshed or if the Apply property is called. For instance, If you need a permanent foreground color for specified folders, you can add a new FileType object like in the following C# sample:

```
EXFILEVIEWLib.FileType fileType = axExFileView1.FileTypes.Add("Temp");
fileType.Folder = true;
fileType.BackColor = ToUInt32(Color.Blue);
fileType.ForeColor = ToUInt32(Color.Yellow);
fileType.Bold = true;
fileType.Apply();
```

The following VFP sample changes the foreground color for the item named "Temp":

```
With thisform.ExFileView1.Get(1).Item("Temp") && AllItems
   .BackColor = RGB(0,0,255)
   .ForeColor = RGB(255,255,255)
   .Bold = .t.
EndWith
```

The BackColor attribute is lost if the control is refreshed or if the Apply property is called. For instance, If you need a permanent foreground color for specified folders, you can add a new FileType object like in the following VFP sample:

```
With thisform.ExFileView1.FileTypes.Add("Temp")
   .Folder = .t.
   .BackColor = RGB(0,0,255)
   .ForeColor = RGB(255,255,255)
   .Bold = .t.
   .Apply()
EndWith
```

property File.FullName as String

Retrieves the full name of the file.

Type	Description
String	A string value that indicates the file/folder's full name.

The FullName property is read-only. Use the FullName property to get the path for the file or folder. Use Folder property to check whether the File object contains a file or a folder. Use the Name to get the displayed file/folder's name. Use the BrowseFolderPath property to specify the path to the browsed folder. Use the Get method to retrieve the selected items. The RelativeName property gets the relative path for the file or folder, based on the ExploreFromHere property. The Created property specifies the date when the file or directory was created. The Size property gets the size of the file in bytes. The Modified property specifies the date when the file was last written to, truncated, or overwritten. The Accessed property specifies the date when the file was last read from, written to, or for executable files, run.

The following VB sample displays the full name for all selected items:

```
With ExFileView1.Get(SelItems)
   Dim i As Long
   For i = 0 To .Count - 1
      With .Item(i)
         Debug.Print .FullName
      End With
   Next
End With
```

The following C++ sample displays the full name for all selected items:

```
CFiles files = m_fileview.GetGet( 0 /*SelItems*/ );
for ( long i = 0; i < files.GetCount(); i++ )
   OutputDebugString( files.GetItem( COleVariant( i ) ).GetFullName() );
```

The following VB.NET sample displays the full name for all selected items:

```
With AxExFileView1.get_Get(EXFILEVIEWLib.TypeEnum.SelItems)
   Dim i As Integer
   For i = 0 To .Count - 1
      Debug.WriteLine(.Item(i).FullName)
```

```
    Next
End With
```

The following C# sample displays the full name for all selected items:

```
EXFILEVIEWLib.Files files = axExFileView1.get_Get(EXFILEVIEWLib.TypeEnum.SelItems);
for (int i = 0; i < files.Count; i++)
    System.Diagnostics.Debug.WriteLine( files[i].FullName );
```

The following VFP sample displays the full name for all selected items:

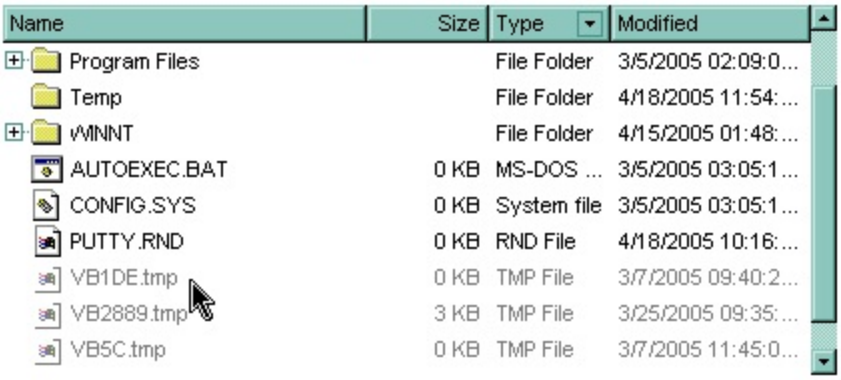
```
With thisform.ExFileView1.Get( 0 ) && SelItems
    local i
    for i = 0 to .Count - 1
        with .Item(i)
            wait window nowait .FullName
        endwith
    next
EndWith
```

property File.Ghosted as Boolean

Returns or sets a value that determines whether a file appears as unavailable (it appears dimmed).

Type	Description
Boolean	A boolean expression that indicates whether an item appears as unavailable.

Use the Ghosted property to change the item's appearance. Use the BackColor and ForeColor properties to change the background and foreground colors for a specified file/folder. Use the Get method to get the collection of files/folders. Use the Folder property to specify whether the File object hosts a file or a folder. Use the BrowseFolderPath property to specify the path to the browsed folder.



The following VB sample marks the tmp files:

```
With ExFileView1.Get(AllItems).Get("*.tmp")
   Dim i As Long
   For i = 0 To .Count - 1
      With .Item(i)
         .Ghosted = True
      End With
   Next
End With
```

The following C++ sample marks the tmp files:

```
CFiles files = m_fileview.GetGet( 1 /*AllItems*/ ).GetGet("*.tmp");
for ( long i = 0; i < files.GetCount(); i++ )
   files.GetItem( COleVariant( i ) ).SetGhosted( TRUE );
```

The following VB.NET sample marks the tmp files:

```
With AxExFileView1.get_Get(EXFILEVIEWLib.TypeEnum.AllItems).Get("*.tmp")
   Dim i As Integer
   For i = 0 To .Count - 1
      .Item(i).Ghosted = True
   Next
End With
```

The following C# sample marks the tmp files:

```
EXFILEVIEWLib.Files files =
axExFileView1.get_Get(EXFILEVIEWLib.TypeEnum.AllItems).get_Get("*.tmp");
for (int i = 0; i < files.Count; i++)
   files[i].Ghosted = true;
```

The following VFP sample marks the tmp files:

```
With thisform.ExFileView1.Get( 1 ).Get("*.tmp") && AllItems
   local i
   for i = 0 to .Count - 1
      with .Item(i)
         .Ghosted = .t.
      endwith
   next
EndWith
```

property File.Modified as Date

Retrieves the date when the file was last written to, truncated, or overwritten.

Type	Description
Date	A DATE expression that specifies the date when the file was last written to, truncated, or overwritten.

The Modified property specifies the date when the file was last written to, truncated, or overwritten. The Size property gets the size of the file in bytes. The FullName property indicates the fully name of the file or directory. The Created property specifies the date when the file or directory was created. The Accessed property specifies the date when the file was last read from, written to, or for executable files, run.

property File.Name as String

Retrieves the file's name.

Type	Description
String	A string expression that specifies the file/folder's name.

The Name property is read-only. Use the Name property to get the file/folder name. Use the FullName property to retrieve the full path for file/folder. Use the Folder property to check if the File object holds a file or a folder. Use the BrowseFolderPath property to specify the path to the browsed folder. Use the Get method to retrieve the selected items. The RelativeName property gets the relative path for the file or folder, based on the ExploreFromHere property. The Created property specifies the date when the file or directory was created. The Size property gets the size of the file in bytes. The Modified property specifies the date when the file was last written to, truncated, or overwritten. The Accessed property specifies the date when the file was last read from, written to, or for executable files, run.

The following VB sample displays the name for all selected items:

```
With ExFileView1.Get(SelItems)
   Dim i As Long
   For i = 0 To .Count - 1
      With .Item(i)
         Debug.Print .FullName
      End With
   Next
End With
```

The following C++ sample displays the name for all selected items:

```
CFiles files = m_fileview.GetGet( 0 /*SelItems*/ );
for ( long i = 0; i < files.GetCount(); i++ )
   OutputDebugString( files.GetItem( COleVariant( i ) ).GetFullName() );
```

The following VB.NET sample displays the name for all selected items:

```
With AxExFileView1.get_Get(EXFILEVIEWLib.TypeEnum.SelItems)
   Dim i As Integer
   For i = 0 To .Count - 1
      Debug.WriteLine(.Item(i).FullName)
```

```
    Next
End With
```

The following C# sample displays the name for all selected items:

```
EXFILEVIEWLib.Files files = axExFileView1.get_Get(EXFILEVIEWLib.TypeEnum.SelItems);
for (int i = 0; i < files.Count; i++)
    System.Diagnostics.Debug.WriteLine( files[i].FullName );
```

The following VFP sample displays the name for all selected items:

```
With thisform.ExFileView1.Get( 0 ) && SellItems
    local i
    for i = 0 to .Count - 1
        with .Item(i)
            wait window nowait .FullName
        endwith
    next
EndWith
```

property File.ParseName as String

Retrieves the file's parse name.

Type	Description
String	A string expression that indicates the object's parse name

The ParseName indicates the name of the object relative to its folder. The ParseName property is read-only. For instance, the name of the drive H: includes its label, while the parsed name is H:\ only. Use the FullName property to retrieve the full path for file/folder. Use the Folder property to check if the File object holds a file or a folder. Use the Get method to retrieve the selected items. The RelativeName property gets the relative path for the file or folder, based on the ExploreFromHere property.

property File.RelativeName as String

Retrieves the relative name of the file.

Type	Description
String	A string value that indicates the file/folder's relative name.

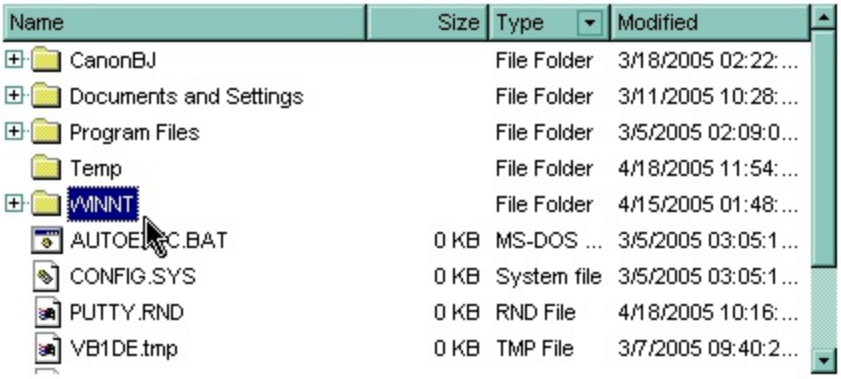
The RelativeName property is read-only. The relative name of the object does not include the folder being referred by the ExploreFromHere property. The FullName property gets the full path for the file or folder. Use Folder property to check whether the File object contains a file or a folder. Use the Name to get the displayed file/folder's name. For instance, if the ExploreFromHere' is 'C:\TEMP' and you browse in that folder to 'C:\TEMP\TEXT\OWN\abc.txt' the RelativeName property returns '\TEXT\OWN\abc.txt', so not the full name and path but the relative path to ExploreFromHere property.

property File.Selected as Boolean

Specifies whether the file is selected or unselected.

Type	Description
Boolean	A boolean expression that indicates whether the file/folder is selected or unselected.

The Selected property specifies whether the file or folder is selected or unselected. The StateChange event is fired, when the user changes the selection. Use the Get property to get all selected files/folders in the current list. Use the BrowseFolderPath property to specify the path to the browsed folder. Use the Folder property to specify whether the File object hosts a file or a folder. Use the SelForeColor and SelBackColor properties to specify the foreground and background colors for selected items. You can use the Expand method to expand the parent if necessary, ensures that the giving path fits the control's area and select it.



The following VB sample selects the "WINNT" folder:

```
With ExFileView1.Get(AllItems).Item("WINNT")
   .Selected = True
End With
```

The following VB enumerates the selected files and folders:

```
Dim i As Long
With ExFileView1.Get(SelItems)
   For i = 0 To .Count - 1
      Debug.Print .Item(i).FullName
   Next
End With
```

The following C++ sample selects the "WINNT" folder:

```
CFile1 file = m_fileview.GetGet( 1 /*AllItems*/ ).GetItem(COleVariant( "WINNT" ));
file.SetSelected( TRUE );
```

The following C++ enumerates the selected files and folders:

```
CFiles files = m_fileview.GetGet( 0 /*SelItems*/ );
for ( long i = 0; i < files.GetCount(); i++ )
   OutputDebugString( files.GetItem( COleVariant( i ) ).GetFullName() );
```

The following VB.NET sample selects the "WINNT" folder:

```
With AxExFileView1.get_Get(EXFILEVIEWLib.TypeEnum.AllItems).Item("WINNT")
   .Selected = True
End With
```

The following VB.NET enumerates the selected files and folders:

```
With AxExFileView1.get_Get(EXFILEVIEWLib.TypeEnum.SelItems)
   Dim i As Integer
   For i = 0 To .Count - 1
      Debug.WriteLine(.Item(i).FullName)
   Next
End With
```

The following C# sample selects the "WINNT" folder:

```
EXFILEVIEWLib.File file = axExFileView1.get_Get(EXFILEVIEWLib.TypeEnum.AllItems)
["WINNT"];
file.Selected = true;
```

The following C# enumerates the selected files and folders:

```
EXFILEVIEWLib.Files files = axExFileView1.get_Get(EXFILEVIEWLib.TypeEnum.SelItems);
for (int i = 0; i < files.Count; i++)
{
   EXFILEVIEWLib.File file = files[i];
   System.Diagnostics.Debug.WriteLine(file.FullName);
}
```

The following VFP sample selects the "WINNT" folder:

```
With thisform.ExFileView1.Get( 1 ).Item("WINNT")
   .Selected = .t.
EndWith
```

The following C# enumerates the selected files and folders:

```
with thisform.ExFileView1.Get( 0 ) && SelItems
   local i
   for i = 0 to .Count - 1
      with .Item(i)
         wait window nowait .FullName
      endwith
   next
 endwith
```

property File.Size as Long

Retrieves the size of the file.

Type	Description
Long	A long expression that specifies the size of the file (in bytes).

The Size property gets the size of the file in bytes. The Size property gets 0, for folder objects. The FullName property indicates the fully name of the file or directory. The Modified property specifies the date when the file was last written to, truncated, or overwritten. The Created property specifies the date when the file or directory was created. The Accessed property specifies the date when the file was last read from, written to, or for executable files, run.

property File.State as ChangeEnum

Indicates the file's changed state.

Type	Description
ChangeEnum	A ChangeEnum expression that indicates the file's state.

The State property specifies if the file/folder was added, removed or changed during the Change event. The Change event is fired when the system notifies the control that there was a change in the browsed folder. Use the BrowseFolderPath property to indicates the browsed folder. The State property is Unchanged, if the State property is called outside of Change event. Use the Folder property to specify whether the object holds information about a folder of a file. Use the Item property to access a file giving its index in the Files collection. Use the Count property to retrieve the number of File objects in the Files collection.

The following VB sample displays the files that have been changed in the browsed folder:

```
Private Sub ExFileView1_Change(ByVal Files As EXFILEVIEWLibCtl.IFiles)
   Dim f As EXFILEVIEWLibCtl.File
   For Each f In Files
      Debug.Print "'" & f.Name & "' " & IIf(f.Folder, "foder", "file") & " " & IIf(f.State = Added,
"added", IIf(f.State = Changed, "changed", IIf(f.State = Deleted, "deleted", "unchanged")))
   Next
End Sub
```

Open a new Windows Explorer instance that browses the same folder as your control. Add new folders, remove folders, or change regular files. Your VB output should look like the following:

```
The 'New Folder (2)' foder - Added
The 'New Folder (2)' foder - Deleted
The 'New Folder' foder was deleted
The 'New Text Document.txt' file was added
The 'New Text Document.txt' file was changed
```

The following C++ sample displays the files that have been changed in the browsed folder:

```
#include "Files.h"
#include "File.h"
void OnChangeExfileview1(LPDISPATCH Files)
```

```
{
    CFiles files( Files ); files.m_bAutoRelease = FALSE;
    for ( long i = 0; i < files.GetCount(); i++ )
    {
        CFile1 file = files.GetItem( COleVariant( long( i ) ) );
        CString strState;
        switch ( file.GetState() )
        {
            case 0:
            {
                strState = "unchanged";
                break;
            }
            case 1:
            {
                strState = "changed";
                break;
            }
            case 2:
            {
                strState = "added";
                break;
            }
            case 3:
            {
                strState = "deleted";
                break;
            }
        }
        CString strOutput;
        strOutput.Format( "'%s' %s %s\n", file.GetName(), (file.GetFolder() ? "folder" : "file" ),
strState );
        OutputDebugString( strOutput );
    }
}
```

The following VB.NET sample displays the files that have been changed in the browsed

folder:

```
Private Sub AxExFileView1_Change(ByVal sender As Object, ByVal e As
AxEXFILEVIEWLib._IExFileViewEvents_ChangeEvent) Handles AxExFileView1.Change
   Dim f As EXFILEVIEWLib.File
   For Each f In e.files
      Debug.WriteLine("'" & f.Name & "' " & IIf(f.Folder, "foder", "file") & " " & IIf(f.State =
EXFILEVIEWLib.ChangeEnum.Added, "added", IIf(f.State =
EXFILEVIEWLib.ChangeEnum.Changed, "changed", IIf(f.State =
EXFILEVIEWLib.ChangeEnum.Deleted, "deleted", "unchanged"))))
   Next
End Sub
```

The following C# sample displays the files that have been changed in the browsed folder:

```
private void axExFileView1_Change(object sender,
AxEXFILEVIEWLib._IExFileViewEvents_ChangeEvent e)
{
   for (int i = 0; i < e.files.Count; i++)
   {
      EXFILEVIEWLib.File file = e.files[i];
      string strOutput = "'" + file.Name + "' ";
      strOutput += (file.Folder ? "folder" : "file") + " ";
      strOutput += (file.State == EXFILEVIEWLib.ChangeEnum.Added ? "added" : (file.State
== EXFILEVIEWLib.ChangeEnum.Deleted ? "deleted" : (file.State ==
EXFILEVIEWLib.ChangeEnum.Changed ? "changed" : "unchanged")));
      System.Diagnostics.Debug.WriteLine(strOutput);
   }
}
```

The following VFP sample displays the files that have been changed in the browsed folder:

```
*** ActiveX Control Event ***
LPARAMETERS files

with files
   local i
   for i = 0 to .Count - 1
```

```
    with .Item(i)
        wait window nowait .Name + " " + str(.State)
    endwith
  next
endwith
```

property File.Type as String

Specifies the file's type.

Type	Description
String	A String expression that indicates the file's type.

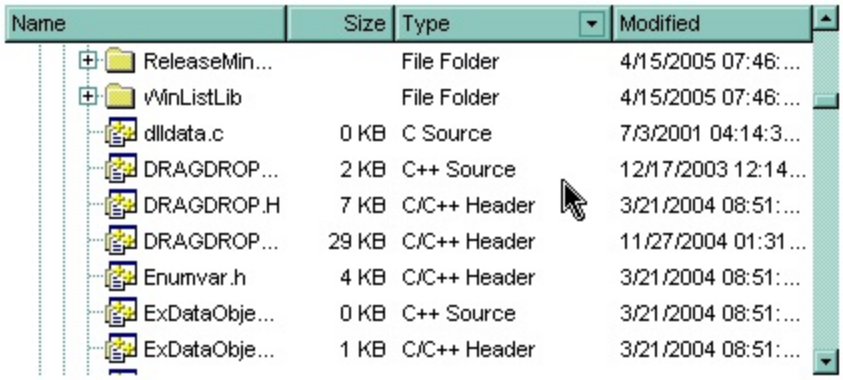
The Type property specifies the type of the file. For instance, the type for a folder object is "File Folder", or for a HTML file it is "HTML Document". The Type property is defined by the system. The Type property does not retrieve the file's extension. In order to get the file's extension you have to use the Name property. The Type column of the control, displays the file types. For instance, the type for *.cpp files is "C++ source", and for *.h files is "C/C++ Header". Even if the Type property is read only you can change the file's type by using Type property of the FileType object, like in the following samples.

The following VB sample changes the string being displayed in the "Type" column, for cpp and h files:

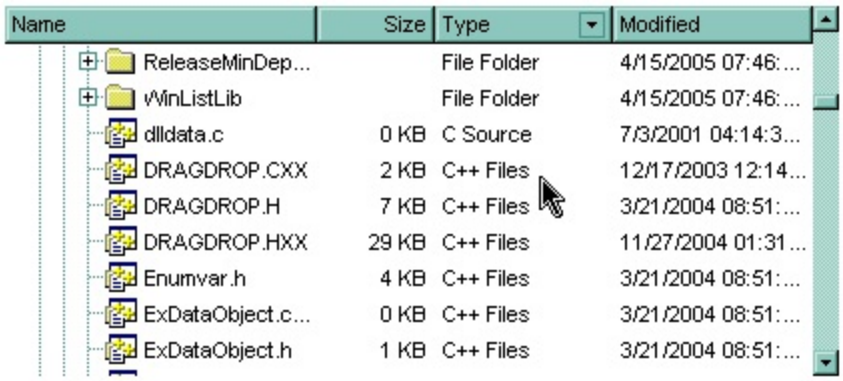
```
With ExFileView1.FileTypes
   .Add("*.cpp *.h").Type = "C++ Files"
   .Apply
End With
```

The s ample specifies that all files with 'cpp' and 'h' extensions should display on Type column the "C++ files" string, instead "C++ Source", and "C/C++ Header".

The screen shot shows the control before running the sample:



The screen shot shows the control after running the sample (as you can see the "C++ Source", "C/C++ Header" strings are replaced with the "C++ Files" string):



The following C++ sample changes the string being displayed in the "Type" column, for cpp and h files:

```
#include "FileType.h"
#include "FileTypes.h"
CFileType fileType = m_fileview.GetFileTypes().Add("*.cpp *.h");
fileType.SetType( "C++ Files" );
fileType.Apply();
```

The following VB.NET sample changes the string being displayed in the "Type" column, for cpp and h files:

```
With AxExFileView1.FileTypes.Add("*.cpp *.h")
   .Type = "C++ Files"
   .Apply()
End With
```

The following C# sample changes the string being displayed in the "Type" column, for cpp and h files:

```
EXFILEVIEWLib.FileType fileType = axExFileView1.FileTypes.Add("*.cpp *.h");
fileType.Type = "C++ Files";
fileType.Apply();
```

The following VFP sample changes the string being displayed in the "Type" column, for cpp and h files:

```
With thisform.ExFileView1.FileTypes.Add("*.cpp *.h")
   .Type = "C++ Files"
   .Apply()
EndWith
```

Files object

The Files object stores a collection of File objects. Use the Get property o the control to access the Files collection. Use the Get property of the Files collection to filter the File objects into a new Files collection. The Files collection implements "for each" statement. Use the Item property to access a File object given its name or its index into collection.

Name	Description
Count	Returns the number of objects in a collection.
Get	Retrieves a new collection that contain files that match the pattern.
Item	Returns a FileType object given its index or its name.

property Files.Count as Long

Returns the number of objects into collection.

Type	Description
Long	A long value that indicates the number of objects into collection.

The Count property gets the number of files in the collection. Use the Item property to retrieve the file or folder giving its index. Use the Get property to get only files that matches giving patterns. Use the Get property to get files and folder being browsed. Use the BrowseFolderPath property to specify the path to the browsed folder. Use the Folder property to specify whether the File object hosts a file or a folder.

The following VB sample displays the list of files as they are displayed:

```
With ExFileView1.Get(VisibleItems)
   For i = 0 To .Count - 1
      With .Item(i)
         Debug.Print .Name
      End With
   Next
End With
```

The following C++ sample displays the list of files as they are displayed:

```
CFiles files = m_fileview.GetGet( 3 /*VisibleItems*/ );
for ( long i = 0; i < files.GetCount(); i++ )
   OutputDebugString( files.GetItem( COleVariant( i ) ).GetName() );
```

The following VB.NET sample displays the list of files as they are displayed:

```
With AxExFileView1.get_Get(EXFILEVIEWLib.TypeEnum.VisibleItems)
   Dim i As Integer
   For i = 0 To .Count - 1
      With .Item(i)
         Debug.WriteLine(.Name())
      End With
   Next
End With
```

The following C# sample displays the list of files as they are displayed:

```
EXFILEVIEWLib.Files files = axExFileView1.get_Get(EXFILEVIEWLib.TypeEnum.VisibleItems);
for (int i = 0; i < files.Count; i++)
{
   EXFILEVIEWLib.File file = files[i];
   System.Diagnostics.Debug.WriteLine(file.Name);
}
```

The following VFP sample displays the list of files as they are displayed:

```
With thisform.ExFileView1.Get(3)   && VisibleItems
   For i = 0 To .Count - 1
      With .Item(i)
         wait window nowait .Name
      EndWith
   Next
EndWith
```

property Files.Get (Pattern as String) as Files

Retrieves a new collection that contain a collection of files that match the pattern.

Type	Description
Pattern as String	A string expression that defines the pattern, like "*.bmp *.jpg"
Files	A Files collection that contains File objects that match the pattern.

Use the Get property to selects only File objects that match a pattern. Use the Get property to get files and folder being browsed. Use the BrowseFolderPath property to specify the path to the browsed folder. Use the FullName property to retrieve the full name of the file or folder. Use the Name property to retrieve the name of the file or folder. Use the Folder property to specify whether the File object hosts a file or a folder.

The following VB sample displays BMP, GIF and JPG files as they are displayed:

```
Dim i As Long
With ExFileView1.Get(VisibleItems).Get("*.bmp *.gif *.jpg")
   For i = 0 To .Count - 1
      Debug.Print .Item(i).FullName
   Next
End With
```

The following C++ sample displays BMP, GIF and JPG files as they are displayed:

```
CFiles files = m_fileview.GetGet( 3 /*VisibleItems*/ ).GetGet( "*.bmp *.gif *.jpg" );
for ( long i = 0; i < files.GetCount(); i++ )
   OutputDebugString( files.GetItem(COleVariant( i ) ).GetFullName() );
```

The following VB.NET sample displays BMP, GIF and JPG files as they are displayed:

```
With AxExFileView1.get_Get(EXFILEVIEWLib.TypeEnum.VisibleItems).Get("*.bmp *.gif
*.jpg")
   Dim i As Integer
   For i = 0 To .Count - 1
      Debug.WriteLine(.Item(i).FullName)
   Next
End With
```

The following C# sample displays BMP, GIF and JPG files as they are displayed:

```
EXFILEVIEWLib.Files files =
axExFileView1.get_Get(EXFILEVIEWLib.TypeEnum.AllItems).get_Get("*.bmp *.gif *.jpg");
for (int i = 0; i < files.Count; i++)
   System.Diagnostics.Debug.WriteLine(files[i].FullName);
```

The following VFP sample displays BMP, GIF and JPG files as they are displayed:

```
local i
With thisform.ExFileView1.Get( 3 ).Get("*.bmp *.gif *.jpg") && VisibleItems
   For i = 0 To .Count - 1
      wait window nowait .Item(i).FullName
   Next
EndWith
```

property Files.Item (Index as Variant) as File

Returns a FileType object given its index or its name.

Type	Description
Index as Variant	A long expression that indicates the index within collection, or a string expression that indicates the filename.
File	A File object being retrieved.

Use the Item property to retrieve the file or folder giving its index. The Count property gets the number of files in the collection. Use the Get property to get only files that matches giving patterns. Use the Get property to get files and folder being browsed. Use the BrowseFolderPath property to specify the path to the browsed folder. Use the Folder property to specify whether the File object hosts a file or a folder.

The following VB sample displays the list of files as they are displayed:

```
With ExFileView1.Get(VisibleItems)
   For i = 0 To .Count - 1
      With .Item(i)
         Debug.Print .Name
      End With
   Next
End With
```

The following C++ sample displays the list of files as they are displayed:

```
CFiles files = m_fileview.GetGet( 3 /*VisibleItems*/ );
for ( long i = 0; i < files.GetCount(); i++ )
   OutputDebugString( files.GetItem( COleVariant( i ) ).GetName() );
```

The following VB.NET sample displays the list of files as they are displayed:

```
With AxExFileView1.get_Get(EXFILEVIEWLib.TypeEnum.VisibleItems)
   Dim i As Integer
   For i = 0 To .Count - 1
      With .Item(i)
         Debug.WriteLine(.Name())
      End With
   Next
```

```
End With
```

The following C# sample displays the list of files as they are displayed:

```
EXFILEVIEWLib.Files files = axExFileView1.get_Get(EXFILEVIEWLib.TypeEnum.VisibleItems);
for (int i = 0; i < files.Count; i++)
{
   EXFILEVIEWLib.File file = files[i];
   System.Diagnostics.Debug.WriteLine(file.Name);
}
```

The following VFP sample displays the list of files as they are displayed:

```
With thisform.ExFileView1.Get(3)   && VisibleItems
   For i = 0 To .Count - 1
      With .Item(i)
         wait window nowait .Name
      EndWith
   Next
EndWith
```

FileType object

The FileType object holds a set of attributes like: font attributes, color, icons, that are applied for files/folders that match the object's Pattern property. You can use any pattern to build a new FileType, like *.bmp *.jpg a*.exe ????.exe and so on. A rule defined by one FileType object is not applied to the current folder, if the Apply method of FileType object, or the Apply method of FileTypes object is not invoked. The FileType' attributes are applied also, if the control is refreshed, or if the user changes the browsed folder. If the FileType object has the From and To set, the FileType object is applied only if the files/folders were changed between From and To dates. In this case, the ExFileView control runs new threads in the background, that looks recursively for any file that matches the FileType's pattern property and was updated in the given interval. When one of this file is found the rule is applied. Using threads the ExFileView control will never lock your application.

Name	Description
Apply	Applies the changes to the current list.
BackColor	Retrieves or sets a value that indicates the background color for the files that match the FileType's pattern.
Bold	Specifies whether the files that match the pattern should appear in bold.
Folder	Retrieves or sets a value indicating whether the changes are applied only for folder objects.
ForeColor	Retrieves or sets a value that indicates the foreground color for the files that match the FileType's pattern.
From	Specifies whether the FileType object is applied for files/folders that have been changed after given date.
HasPattern	Specifies whether the FileType's pattern contains wild characters. If the FileType's pattern contains no wild cards, the pattern defines exactly the file name.
IconIndex	Indicates the icon index used to replace the default icon for files that match the FileType's pattern.
Italic	Specifies whether the files that match the pattern should appear in italic.
Pattern	Specifies the pattern that defines the files into a group, like '*.cpp *h'.
StrikeOut	Specifies whether the files that match the pattern should appear in strikeout.
To	Specifies whether the FileType object is applied for files/folders that have been changed before give date.

Type	Retrieves or sets a value that indicates the string gets displayed on the Type column.
Underline	Specifies whether the files that match the pattern appear as underlined.

method FileType.Apply ()

Applies the changes to the current list.

Type	Description

After you define the FileType object you have to call Apply method, in order to reflect the changes to the current list. Use the BrowseFolderPath property to specify the browsed folder. The Add method does **not** invoke the Apply method. If your are going to add more FileType objects to the FileTypes collection, it is not necessary to call Apply method for each new FileType, instead you can call Apply method of FileTypes collection.

The following VB sample bolds the cpp and h files:

```
With ExFileView1.FileTypes.Add("*.cpp *.h")
   .Bold = True
   .Apply
End With
```

The following C++ sample bolds the cpp and h files:

```
#include "FileType.h"
#include "FileTypes.h"
CFileType fileType = m_fileview.GetFileTypes().Add("*.cpp *.h");
fileType.SetBold( TRUE );
fileType.Apply();
```

The following VB.NET sample bolds the cpp and h files:

```
With AxExFileView1.FileTypes.Add("*.cpp *.h")
   .Bold = True
   .Apply()
End With
```

The following C# sample bolds the cpp and h files:

```
EXFILEVIEWLib.FileType fileType = axExFileView1.FileTypes.Add("*.cpp *.h");
fileType.Bold = true;
fileType.Apply();
```

The following VFP sample bolds the cpp and h files:

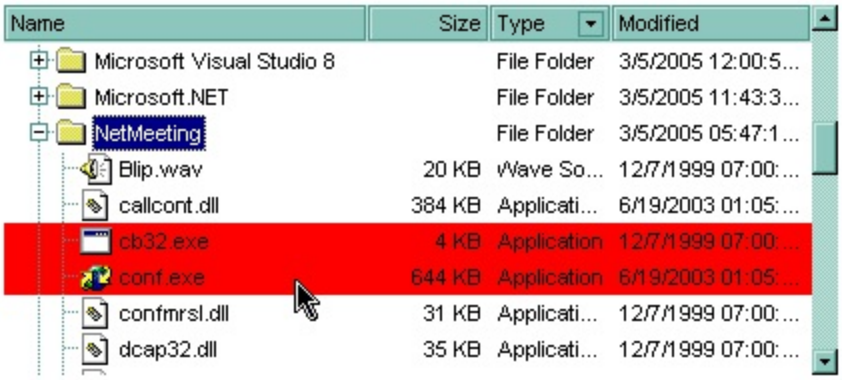
```
With thisform.ExFileView1.FileTypes.Add("*.cpp *.h")
    .Bold = .t.
    .Apply()
EndWith
```

property FileType.BackColor as Color

Retrieves or sets a value that indicates the background color for the files that match the FileType's pattern.

Type	Description
Color	A color expression that defines the background color used to paint the items that match the Pattern.

Use the BackColor and ForeColor properties to color your items that match a pattern. The Apply method applies the changes to the current list. Use the BrowseFolderPath property to specify the path to the browsed folder. Use the BackColor property to specify the control's background color. Use the BackColor property to specify the file/folder's background color.



The following VB sample changes the background color for the exe files:

```
With ExFileView1.FileTypes
   .Add("*.exe").BackColor = vbRed
   .Apply
End With
```

The following C++ sample changes the background color for the exe files:

```
#include "FileType.h"
#include "FileTypes.h"
CFileType fileType = m_fileview.GetFileTypes().Add("*.exe");
fileType.SetBackColor( RGB(255,0,0) );
fileType.Apply();
```

The following VB.NET sample changes the background color for the exe files:

```
With AxExFileView1.FileTypes.Add("*.exe")
```

```
    .BackColor = ToUInt32(Color.Red)
    .Apply()
End With
```

where the ToUInt32 function converts a Color expression to OLE_COLOR type,

```
Shared Function ToUInt32(ByVal c As Color) As UInt32
    Dim i As Long
    i = c.R
    i = i + 256 * c.G
    i = i + 256 * 256 * c.B
    ToUInt32 = Convert.ToUInt32(i)
End Function
```

The following C# sample changes the background color for the exe files:

```
EXFILEVIEWLib.FileType fileType = axExFileView1.FileTypes.Add("*.exe");
fileType.BackColor = ToUInt32(Color.Red);
fileType.Apply();
```

where the ToUInt32 function converts a Color expression to OLE_COLOR type,

```
private UInt32 ToUInt32(Color c)
{
    long i;
    i = c.R;
    i = i + 256 * c.G;
    i = i + 256 * 256 * c.B;
    return Convert.ToUInt32(i);
}
```

The following VFP sample changes the background color for the exe files:

```
With thisform.ExFileView1.Add("*.exe")
    .BackColor = RGB(255,0,0)
    .Apply()
EndWith
```

property FileType.Bold as Boolean

Retrieves or sets a value that specifies whether the files that match the FileType's pattern should appear in bold.

Type	Description
Boolean	A boolean expression that indicates whether the files that match the FileType's pattern should appear in bold.

The Bold property specifies whether the file or the folder should appear in bold. Use the Font property to change the control's font. The Apply method applies the changes to the current list. Use the BrowseFolderPath property to specify the path to the browsed folder.

The following VB sample bolds the cpp and h files:

```
With ExFileView1.FileTypes.Add("*.cpp *.h")
   .Bold = True
   .Apply
End With
```

The following C++ sample bolds the cpp and h files:

```
#include "FileType.h"
#include "FileTypes.h"
CFileType fileType = m_fileview.GetFileTypes().Add("*.cpp *.h");
fileType.SetBold( TRUE );
fileType.Apply();
```

The following VB.NET sample bolds the cpp and h files:

```
With AxExFileView1.FileTypes.Add("*.cpp *.h")
   .Bold = True
   .Apply()
End With
```

The following C# sample bolds the cpp and h files:

```
EXFILEVIEWLib.FileType fileType = axExFileView1.FileTypes.Add("*.cpp *.h");
fileType.Bold = true;
fileType.Apply();
```

The following VFP sample bolds the cpp and h files:

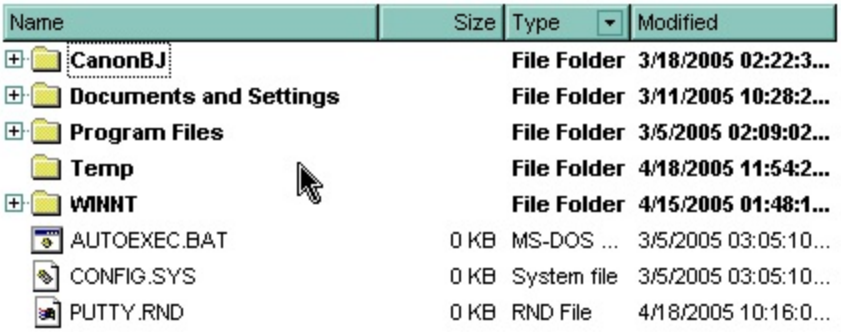
```
With thisform.ExFileView1.FileTypes.Add("*.cpp *.h")
   .Bold = .t.
   .Apply()
EndWith
```

property FileType.Folder as Boolean

Retrieves or sets a value indicating whether the changes are applied only for folder objects.

Type	Description
Boolean	A boolean expression indicating whether the changes are applied only for folder objects.

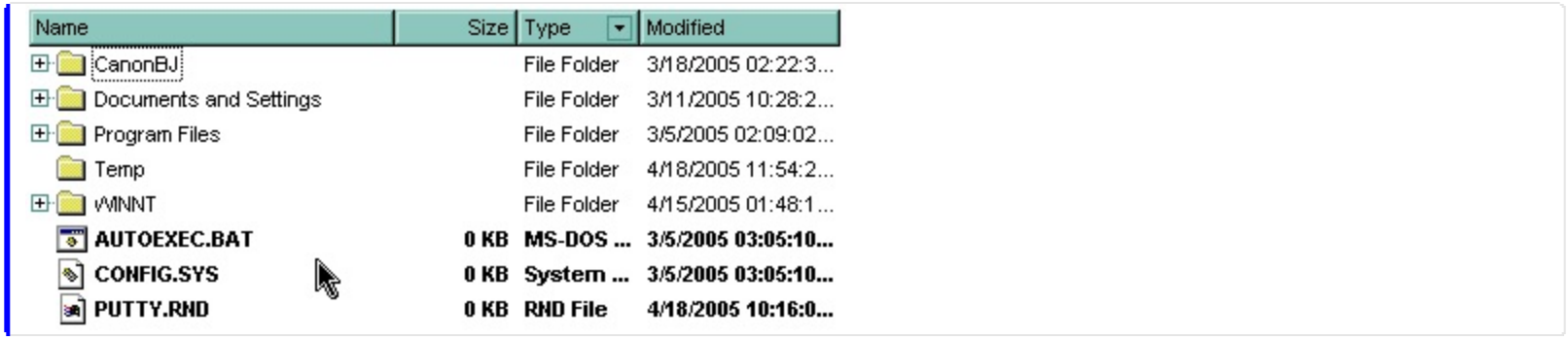
By default, the Folder property is False. If the Folder property is True, the FileType' attributes are applied only for folders, and if the Folder is False the FileType' attributes are applied only for files. The Apply method applies the changes to the current list. Use the BrowseFolderPath property to specify the path to the browsed folder.



The following VB sample bolds all folders:

```
With ExFileView1.FileTypes
   With .Add("*")
      .Folder = True
      .Bold = True
   End With
   .Apply
End With
```

The following screen shot bolds all files (Folder property is False)



The following C++ sample bolds all folders:

```
#include "FileType.h"
#include "FileTypes.h"
CFileType fileType = m_fileview.GetFileTypes().Add("*");
fileType.SetFolder( TRUE );
fileType.SetBold( TRUE );
fileType.Apply();
```

The following VB.NET sample bolds all folders:

```
With AxExFileView1.FileTypes.Add("*")
   .Folder = True
   .Bold = True
   .Apply()
End With
```

The following C# sample bolds all folders:

```
EXFILEVIEWLib.FileType fileType = axExFileView1.FileTypes.Add("*");
fileType.Folder = true;
fileType.Bold = true;
fileType.Apply();
```

The following VFP sample bolds all folders:

```
With thisform.ExFileView1.Add("*")
   .Folder = .t.
   .Bold = .t.
   .Apply()
EndWith
```

property FileType.ForeColor as Color

Retrieves or sets a value that indicates the foreground color for the files that match the FileType's pattern.

Type	Description
Color	A color expression that defines the foreground color for the files that match the FileType's pattern.

Use the BackColor and ForeColor properties to color your items that match a pattern. The Apply method applies the changes to the current list. Use the BrowseFolderPath property to specify the path to the browsed folder. Use the ForeColor property to specify the control's foreground color. Use the ForeColor property to specify the file/folder's foreground color.

The following VB sample changes the foreground color for the exe files:

```
With ExFileView1.FileTypes
   .Add("*.exe").ForeColor = vbRed
   .Apply
End With
```

The following C++ sample changes the foreground color for the exe files:

```
#include "FileType.h"
#include "FileTypes.h"
CFileType fileType = m_fileview.GetFileTypes().Add("*.exe");
fileType.SetForeColor( RGB(255,0,0) );
fileType.Apply();
```

The following VB.NET sample changes the foreground color for the exe files:

```
With AxExFileView1.FileTypes.Add("*.exe")
   .ForeColor = ToUInt32(Color.Red)
   .Apply()
End With
```

where the ToUInt32 function converts a Color expression to OLE_COLOR type,

```
Shared Function ToUInt32(ByVal c As Color) As UInt32
   Dim i As Long
```

```
    i = c.R
    i = i + 256 * c.G
    i = i + 256 * 256 * c.B
    ToUInt32 = Convert.ToUInt32(i)
End Function
```

The following C# sample changes the foreground color for the exe files:

```
EXFILEVIEWLib.FileType fileType = axExFileView1.FileTypes.Add("*.exe");
fileType.ForeColor = ToUInt32(Color.Red);
fileType.Apply();
```

where the ToUInt32 function converts a Color expression to OLE_COLOR type,

```
private UInt32 ToUInt32(Color c)
{
    long i;
    i = c.R;
    i = i + 256 * c.G;
    i = i + 256 * 256 * c.B;
    return Convert.ToUInt32(i);
}
```

The following VFP sample changes the foreground color for the exe files:

```
With thisform.ExFileView1.Add("*.exe")
    .ForeColor = RGB(255,0,0)
    .Apply()
EndWith
```

property FileType.From as Date

Specifies whether the FileType object is applied for files/folders that have been updated after given date.

Type	Description
Date	A date expression that specifies that the FileType' attributes are applied only for files that were updated after the giving date.

The From and To properties define a date interval. The FileType' attributes are applied only for files that match the Pattern property and were updated in the given interval. If the current list contains folders, the control runs a new thread for each folder in order to look recursively for any file that match the pattern and were updated in the interval defined by the From and To properties. If the From and To properties are set to 0, the control doesn't run the any thread. The From and To properties are useful to highlist the files and folder that contains files that were updated into a given interval. The Apply method applies the changes to the current list. Use the BrowseFolderPath property to specify the path to the browsed folder. Use the ModifiedDaysAgo property to specify the string being displayed in the "Modified" column.

The following VB sample highlights the files and folders that were updated yesterday with blue color, and updated today, with red color:

```
With ExFileView1.FileTypes
   With .Add("*")
      .From = Date - 1
      .To = Date
      .ForeColor = vbBlue
   End With
   With .Add("*")
      .From = Date
      .To = .From
      .Bold = True
      .ForeColor = vbRed
   End With
   .Apply
End With
```

The following C++ sample highlights the files and folders that were updated yesterday with blue color, and updated today, with red color:

```
#include "FileType.h"
#include "FileTypes.h"
DATE date = COleDateTime::GetCurrentTime().operator DATE();
CFileType fileType = m_fileview.GetFileTypes().Add("*");
fileType.SetFrom( date - 1 );
fileType.SetTo( date );
fileType.SetForeColor( RGB(0,0,255) );

fileType = m_fileview.GetFileTypes().Add("*");
fileType.SetFrom( date );
fileType.SetTo( fileType.GetFrom() );
fileType.SetBold( TRUE );
fileType.SetForeColor( RGB(255,0,0) );

m_fileview.GetFileTypes().Apply();
```

The following VB.NET sample highlights the files and folders that were updated yesterday with blue color, and updated today, with red color:

```
With AxExFileView1.FileTypes
   With .Add("*")
      .From = Date.Today.AddDays(-1)
      .To = Date.Today
      .ForeColor = ToUInt32(Color.Blue)
   End With
   With .Add("*")
      .From = Date.Today
      .To = .From
      .Bold = True
      .ForeColor = ToUInt32(Color.Red)
   End With
   .Apply()
End With
```

The following C# sample highlights the files and folders that were updated yesterday with blue color, and updated today, with red color:

```
EXFILEVIEWLib.FileType fileType = axExFileView1.FileTypes.Add("*");
```

```
fileType.From = DateTime.Today.AddDays(-1);
fileType.To = DateTime.Today;
fileType.ForeColor = ToUInt32(Color.Blue);
fileType.Bold = true;

fileType = axExFileView1.FileTypes.Add("*");
fileType.From = DateTime.Today;
fileType.To = DateTime.Today;
fileType.ForeColor = ToUInt32(Color.Red);

axExFileView1.FileTypes.Apply();
```

The following VFP sample highlights the files and folders that were updated yesterday with blue color, and updated today, with red color:

```
With thisform.ExFileView1.FileTypes
   With .Add("*")
      .From = Date() - 1
      .To = Date()
      .ForeColor = RGB(0,0,255)
   EndWith
   With .Add("*")
      .From = Date()
      .To = .From
      .Bold = .t.
      .ForeColor = RGB(255,0,0)
   EndWith
   .Apply
EndWith
```

property FileType.HasPattern as Boolean

Specifies whether the FileType's pattern contains wild characters.

Type	Description
Boolean	A boolean expression that specifies whether the FileType's pattern contains wild characters.

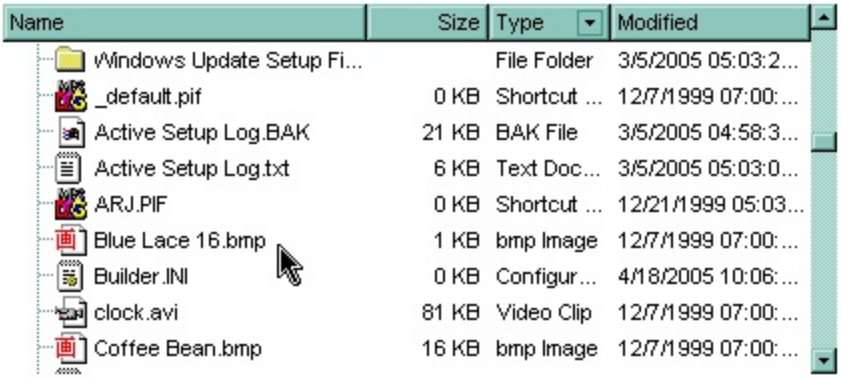
By default the HasPattern property is True. If the FileType's pattern contains no wild cards, the pattern defines exactly the file name. Use the HasPattern property to specify whether the Pattern property contains wild cards like: * or ?. The Apply method applies the changes to the current list. Use the BrowseFolderPath property to specify the path to the browsed folder.

property FileType.IconIndex as Long

Indicates the icon index used to replace the default icon for files that match the FileType's pattern.

Type	Description
Long	A long expression that indicates the icon's key that replace the default icon for files that match the FileType's pattern.

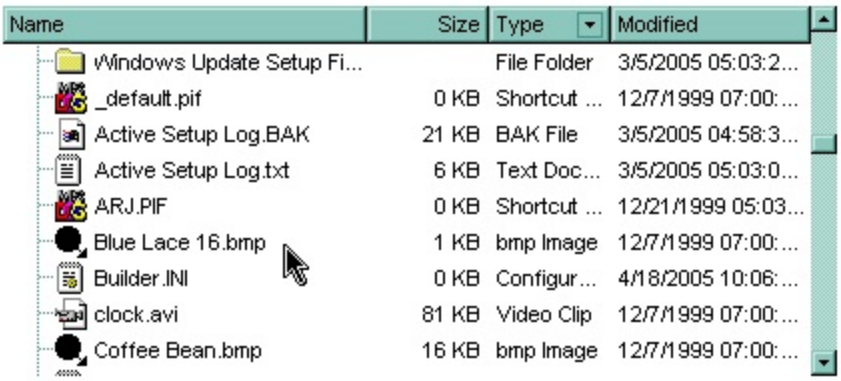
Use the LoadIcons and LoadIcon properties to add new icons to the control. When you are using LoadIcon property to add a new icon, you have to use the icon's key for the IconIndex property. If you are loading a collection of icons using LoadIcons property, you have to use the index of icon into the icons collection. The Apply method applies the changes to the current list. Use the BrowseFolderPath property to specify the path to the browsed folder. The control fires the StateChange event when the user changes the browsed path.



The following VB sample replaces the default icon for files of BMP and JPG types with the icon:

```
With ExFileView1
   .LoadIcon LoadPicture("C:\Temp\sample.ico").Handle, 1234
   With .FileTypes.Add("*.bmp *.jpg")
      .IconIndex = 1234
      .Apply
   End With
End With
```

After running the sample the default icons for BMP and JPG files is changed like:



The following C++ sample replaces the default icon for files of BMP and JPG types:

```
IPictureDisp* pPicture = NULL;
if ( LoadPicture( "c:\\temp\\sample.ico", &pPicture ) )
{
   OLE_HANDLE hIcon = NULL;
   if ( CComQIPtr<IPicture> spPicture( pPicture ) )
      spPicture->get_Handle( &hIcon );
   m_fileview.LoadIcon( hIcon, 1234 );

   CFileType fileType = m_fileview.GetFileTypes().Add("*.bmp *.jpg");
   fileType.SetIconIndex( 1234 );
   fileType.Apply();
}
```

where the LoadPicture function loads a picture from a file, and gets the IPictureDisp interface:

```
#include
BOOL LoadPicture( LPCTSTR szFileName, IPictureDisp** ppPictureDisp )
{
   BOOL bResult = FALSE;
   if ( szFileName )
   {
      OFSTRUCT of;
      HANDLE hFile = NULL;;
#ifdef _UNICODE
      USES_CONVERSION;
      if ( (hFile = (HANDLE)OpenFile( W2A(szFileName), &of;, OF_READ |
OF_SHARE_COMPAT)) != (HANDLE)HFILE_ERROR )
```

```
#else
      if ( (hFile = (HANDLE)OpenFile( szFileName, &of;, OF_READ | OF_SHARE_COMPAT)) !=
(HANDLE)HFILE_ERROR )
#endif
      {
         *ppPictureDisp = NULL;
         DWORD dwHighWord = NULL, dwSizeLow = GetFileSize( hFile, &dwHighWord; );
         DWORD dwFileSize = dwSizeLow;
         HRESULT hResult = NULL;
         if ( HGLOBAL hGlobal = GlobalAlloc(GMEM_MOVEABLE, dwFileSize) )
            if ( void* pvData = GlobalLock( hGlobal ) )
            {
               DWORD dwReadBytes = NULL;
               BOOL bRead = ReadFile( hFile, pvData, dwFileSize, &dwReadBytes;, NULL );
               GlobalUnlock( hGlobal );
               if ( bRead )
               {
                  CComPtr spStream;
                  _ASSERTE( dwFileSize == dwReadBytes );
                  if ( SUCCEEDED( CreateStreamOnHGlobal( hGlobal, TRUE, &spStream;) ) )
                     if ( SUCCEEDED( hResult = OleLoadPicture( spStream, 0, FALSE,
IID_IPictureDisp, (void**)ppPictureDisp ) ) )
                        bResult = TRUE;
               }
            }
         CloseHandle( hFile );
      }
   }
   return bResult;
}
```

The following VB.NET sample replaces the default icon for files of BMP and JPG types:

```
With AxExFileView1
   Dim spPicture As stdole.IPictureDisp =
IPDH.GetIPictureDisp(Image.FromFile("c:\temp\sample.ico"))
   .LoadIcon(spPicture.Handle, 1234)
```

```
    With .FileTypes.Add("*.bmp *.jpg")
        .IconIndex = 1234
        .Apply()
    End With
End With
```

where the IPDH class is defined like follows:

```
Public Class IPDH
    Inherits System.Windows.Forms.AxHost

    Sub New()
        MyBase.New("")
    End Sub

    Public Shared Function GetIPictureDisp(ByVal image As Image) As Object
        GetIPictureDisp = AxHost.GetIPictureDispFromPicture(image)
    End Function

End Class
```

The following C# sample replaces the default icon for files of BMP and JPG types:

```
stdole.IPictureDisp spPicture =
IPDH.GetIPictureDisp(Image.FromFile("c:\\temp\\sample.ico")) as stdole.IPictureDisp;
axExFileView1.LoadIcon( spPicture.Handle, 1234);
EXFILEVIEWLib.FileType fileType = axExFileView1.FileTypes.Add("*.bmp *.jpg");
fileType.IconIndex = 1234;
fileType.Apply();
```

where the IPDH class is defined like follows:

```
internal class IPDH : System.Windows.Forms.AxHost
{
    public IPDH() : base("")
    {
    }

    public static object GetIPictureDisp(System.Drawing.Image image)
```

```
    {
        return System.Windows.Forms.AxHost.GetIPictureDispFromPicture( image );
    }
}
```

The following VFP sample replaces the default icon for files of BMP and JPG types:

```
With thisform.ExFileView1
    local i
    with LoadPicture("C:\temp\sample.ico")
        i = .Handle()
    endwith
    .Object.LoadIcon(i, 1234)
    With .FileTypes.Add("*.bmp *.jpg")
        .IconIndex = 1234
        .Apply
    EndWith
EndWith
```

```
}
```

property FileType.Italic as Boolean

Retrieves or sets a value that specifies whether the files that match the FileType's pattern should appear in italic.

Type	Description
Boolean	A boolean expression that indicates whether files/folders that match the FileType's pattern should appear in italic.

The Italic property specifies whether the file or the folder should appear in italic. Use the Font property to change the control's font. The Apply method applies the changes to the current list. Use the BrowseFolderPath property to specify the path to the browsed folder.

The following VB sample makes the cpp and h files appear in italic:

```
With ExFileView1.FileTypes.Add("*.cpp *.h")
   .Italic = True
   .Apply
End With
```

The following C++ sample makes the cpp and h files appear in italic:

```
#include "FileType.h"
#include "FileTypes.h"
CFileType fileType = m_fileview.GetFileTypes().Add("*.cpp *.h");
fileType.SetItalic( TRUE );
fileType.Apply();
```

The following VB.NET sample makes the cpp and h files appear in italic:

```
With AxExFileView1.FileTypes.Add("*.cpp *.h")
   .Italic = True
   .Apply()
End With
```

The following C# sample makes the cpp and h files appear in italic:

```
EXFILEVIEWLib.FileType fileType = axExFileView1.FileTypes.Add("*.cpp *.h");
fileType.Italic = true;
fileType.Apply();
```

The following VFP sample makes the cpp and h files appear in italic:

```
With thisform.ExFileView1.FileTypes.Add("*.cpp *.h")
    .Italic = .t.
    .Apply()
EndWith
```

property FileType.Pattern as String

Specifies the pattern that defines the files into a group, like '*.cpp *h'.

Type	Description
String	A string expression that defines the pattern to include files based on the rule.

Use the Add property to add a new FileTypes object to the collection, and to specify the group's pattern. The Pattern property specifies a string expression that may include wild cards like * and ?, and defines the files whose name matches the pattern, if the HasPattern property is True. The control uses the Name property to check the pattern. For instance, the *.BMP defines all files that has the extension BMP. The Apply method applies the changes to the current list. Use the BrowseFolderPath property to specify the path to the browsed folder.

property FileType.StrikeOut as Boolean

Retrieves or sets a value that specifies whether the files that match the FileType's pattern should appear in strikeout.

Type	Description
Boolean	A boolean expression that specifies whether the files that match the FileType's pattern should appear in strikeout.

The StrikeOut property specifies whether the file's font is displayed with a horizontal line through. Use the Font property to change the control's font. The Apply method applies the changes to the current list. Use the BrowseFolderPath property to specify the path to the browsed folder.

The following VB sample draws the cpp and h files with a horizontal line through:

```
With ExFileView1.FileTypes.Add("*.cpp *.h")
   .StrikeOut = True
   .Apply
End With
```

The following C++ sample draws the cpp and h files with a horizontal line through:

```
#include "FileType.h"
#include "FileTypes.h"
CFileType fileType = m_fileview.GetFileTypes().Add("*.cpp *.h");
fileType.SetStrikeOut( TRUE );
fileType.Apply();
```

The following VB.NET sample draws the cpp and h files with a horizontal line through:

```
With AxExFileView1.FileTypes.Add("*.cpp *.h")
   .StrikeOut = True
   .Apply()
End With
```

The following C# sample draws the cpp and h files with a horizontal line through:

```
EXFILEVIEWLib.FileType fileType = axExFileView1.FileTypes.Add("*.cpp *.h");
fileType.StrikeOut = true;
fileType.Apply();
```

The following VFP sample draws the cpp and h files with a horizontal line through:

```
With thisform.ExFileView1.FileTypes.Add("*.cpp *.h")
    .StrikeOut = .t.
    .Apply()
EndWith
```

property FileType.To as Date

Specifies whether the FileType object is applied for files/folders that have been updated before give date.

Type	Description
Date	A date expression that specifies that the FileType' attributes are applied only for files that were updated before giving date.

The From and To properties define a date interval. The FileType' attributes are applied only for files that match the Pattern property and were updated in the interval. If the current list contains folders, the control runs a new thread for each folder in order to look recursively for any file that match the pattern and were updated in the interval defined by the From and To properties. If the From and To properties are set to 0, the control doesn't run the any thread. The From and To properties are useful to highlight the files and folder that contains files that were updated into a given interval. The Apply method applies the changes to the current list. Use the BrowseFolderPath property to specify the path to the browsed folder. Use the ModifiedDaysAgo property to specify the string being displayed in the "Modified" column.

The following VB sample highlights the files and folders that were updated yesterday with blue color, and updated today, with red color:

```
With ExFileView1.FileTypes
   With .Add("*")
      .From = Date - 1
      .To = Date
      .ForeColor = vbBlue
   End With
   With .Add("*")
      .From = Date
      .To = .From
      .Bold = True
      .ForeColor = vbRed
   End With
   .Apply
End With
```

The following C++ sample highlights the files and folders that were updated yesterday with blue color, and updated today, with red color:

```
#include "FileType.h"
#include "FileTypes.h"
DATE date = COleDateTime::GetCurrentTime().operator DATE();
CFileType fileType = m_fileview.GetFileTypes().Add("*");
fileType.SetFrom( date - 1 );
fileType.SetTo( date );
fileType.SetForeColor( RGB(0,0,255) );

fileType = m_fileview.GetFileTypes().Add("*");
fileType.SetFrom( date );
fileType.SetTo( fileType.GetFrom() );
fileType.SetBold( TRUE );
fileType.SetForeColor( RGB(255,0,0) );

m_fileview.GetFileTypes().Apply();
```

The following VB.NET sample highlights the files and folders that were updated yesterday with blue color, and updated today, with red color:

```
With AxExFileView1.FileTypes
   With .Add("*")
      .From = Date.Today.AddDays(-1)
      .To = Date.Today
      .ForeColor = ToUInt32(Color.Blue)
   End With
   With .Add("*")
      .From = Date.Today
      .To = .From
      .Bold = True
      .ForeColor = ToUInt32(Color.Red)
   End With
   .Apply()
End With
```

The following C# sample highlights the files and folders that were updated yesterday with blue color, and updated today, with red color:

```
EXFILEVIEWLib.FileType fileType = axExFileView1.FileTypes.Add("*");
```

```
fileType.From = DateTime.Today.AddDays(-1);
fileType.To = DateTime.Today;
fileType.ForeColor = ToUInt32(Color.Blue);
fileType.Bold = true;

fileType = axExFileView1.FileTypes.Add("*");
fileType.From = DateTime.Today;
fileType.To = DateTime.Today;
fileType.ForeColor = ToUInt32(Color.Red);

axExFileView1.FileTypes.Apply();
```

The following VFP sample highlights the files and folders that were updated yesterday with blue color, and updated today, with red color:

```
With thisform.ExFileView1.FileTypes
   With .Add("*")
      .From = Date() - 1
      .To = Date()
      .ForeColor = RGB(0,0,255)
   EndWith
   With .Add("*")
      .From = Date()
      .To = .From
      .Bold = .t.
      .ForeColor = RGB(255,0,0)
   EndWith
   .Apply
EndWith
```

property FileType.Type as String

Retrieves or sets a value that indicates the string gets displayed in the "Type" column.

Type	Description
String	A string expression that indicates the type displayed in the "Type" column.

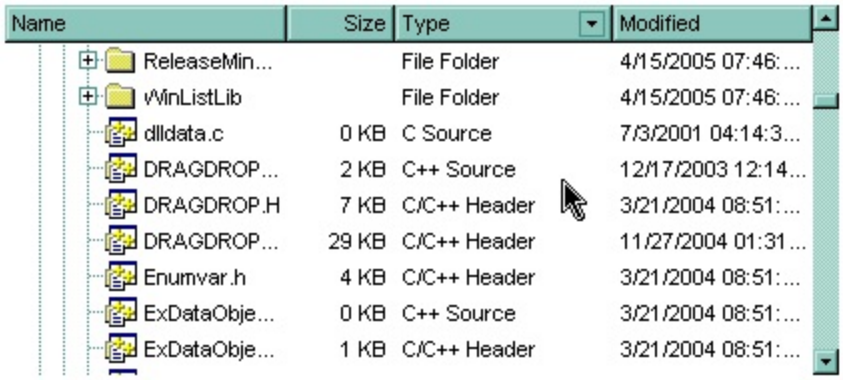
Use the Type property to change the string being displayed in the "Type" column. Use the Type property to get the file/folder's type. The Apply method applies the changes to the current list. Use the BrowseFolderPath property to specify the path to the browsed folder.

The following VB sample changes the string being displayed in the "Type" column, for cpp and h files:

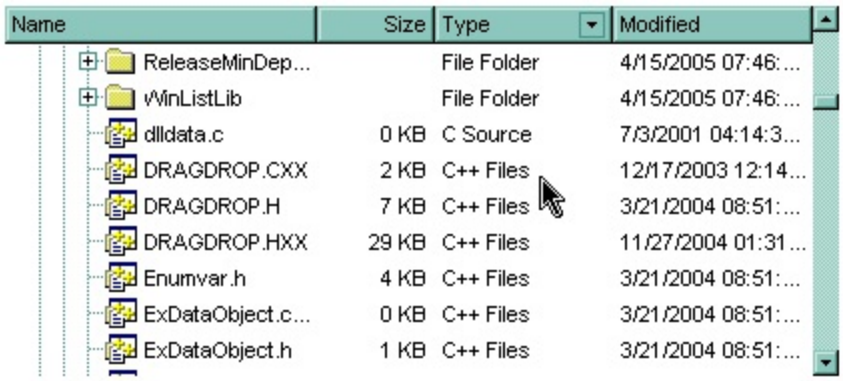
```
With ExFileView1.FileTypes
   .Add("*.cpp *.h").Type = "C++ Files"
   .Apply
End With
```

The s ample specifies that all files with 'cpp' and 'h' extensions should display on Type column the "C++ files" string, instead "C++ Source", and "C/C++ Header".

The screen shot shows the control before running the sample:



The screen shot shows the control after running the sample (as you can see the "C++ Source", "C/C++ Header" strings are replaced with the "C++ Files" string):



The following C++ sample changes the string being displayed in the "Type" column, for cpp and h files:

```
#include "FileType.h"
#include "FileTypes.h"
CFileType fileType = m_fileview.GetFileTypes().Add("*.cpp *.h");
fileType.SetType( "C++ Files" );
fileType.Apply();
```

The following VB.NET sample changes the string being displayed in the "Type" column, for cpp and h files:

```
With AxExFileView1.FileTypes.Add("*.cpp *.h")
   .Type = "C++ Files"
   .Apply()
End With
```

The following C# sample changes the string being displayed in the "Type" column, for cpp and h files:

```
EXFILEVIEWLib.FileType fileType = axExFileView1.FileTypes.Add("*.cpp *.h");
fileType.Type = "C++ Files";
fileType.Apply();
```

The following VFP sample changes the string being displayed in the "Type" column, for cpp and h files:

```
With thisform.ExFileView1.FileTypes.Add("*.cpp *.h")
   .Type = "C++ Files"
   .Apply()
EndWith
```

property FileType.Underline as Boolean

Retrieves or sets a value that specifies whether the files that match the FileType's pattern appear as underlined.

Type	Description
Boolean	A boolean expression that specifies whether the files that match the FileType's pattern appear as underlined.

The Underline property specifies whether the file or the folder is underlined. Use the Font property to change the control's font. The Apply method applies the changes to the current list. Use the BrowseFolderPath property to specify the path to the browsed folder.

The following VB sample underlines the cpp and h files:

```
With ExFileView1.FileTypes.Add("*.cpp *.h")
   .Underline = True
   .Apply
End With
```

The following C++ sample underlines the cpp and h files:

```
#include "FileType.h"
#include "FileTypes.h"
CFileType fileType = m_fileview.GetFileTypes().Add("*.cpp *.h");
fileType.SetUnderline( TRUE );
fileType.Apply();
```

The following VB.NET sample underlines the cpp and h files:

```
With AxExFileView1.FileTypes.Add("*.cpp *.h")
   .Underline = True
   .Apply()
End With
```

The following C# sample underlines the cpp and h files:

```
EXFILEVIEWLib.FileType fileType = axExFileView1.FileTypes.Add("*.cpp *.h");
fileType.Underline = true;
fileType.Apply();
```

The following VFP sample underlines the cpp and h files:

```
With thisform.ExFileView1.FileTypes.Add("*.cpp *.h")
    .Underline = .t.
    .Apply()
EndWith
```

FileTypes object

The FileTypes object contains a collection of FileType objects. The FileTypes collection contains rule of changes that are applied to the current list. Here's the list of supported properties and methods:

Name	Description
Add	Adds a FileType object to the collection and returns a reference to the newly created object.
Apply	Applies the changes to the current list.
Clear	Clears all the elements of collection.
Count	Returns the number of objects in a collection.
Item	Retrieves a FileType object given its index into collection.
Remove	Removes a specific member from the FileTypes collection.

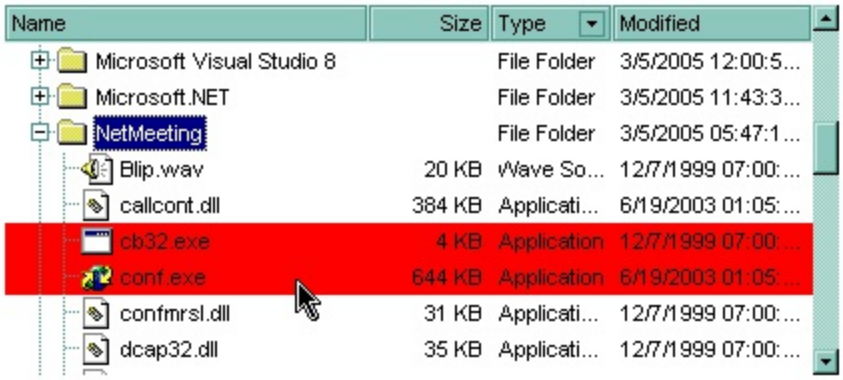
method FileTypes.Add (Pattern as String)

Adds a FileType object to the collection and returns a reference to the newly created object.

Type	Description
Pattern as String	A string expression that may include wild cards like * or ?.

Return	Description
FileType	A FileType object that has been created.

Use the Add property to change the appearance for specified files or folders. Use the FileTypes property to access the FileType objects collection. The new appearance is applied even if the user changes the browsed folder. The BrowseFolderPath property specifies the path to the browsed folder. Use the Apply method to apply all changes to the current list, else they will take effect as soon as the user browses a new folder.



The following VB sample changes the background color for the exe files:

```
With ExFileView1.FileTypes
   .Add("*.exe").BackColor = vbRed
   .Apply
End With
```

The following C++ sample changes the background color for the exe files:

```
#include "FileType.h"
#include "FileTypes.h"
CFileType fileType = m_fileview.GetFileTypes().Add("*.exe");
fileType.SetBackColor( RGB(255,0,0) );
fileType.Apply();
```

The following VB.NET sample changes the background color for the exe files:

```
With AxExFileView1.FileTypes.Add("*.exe")
   .BackColor = ToUInt32(Color.Red)
   .Apply()
End With
```

where the ToUInt32 function converts a Color expression to OLE_COLOR type,

```
Shared Function ToUInt32(ByVal c As Color) As UInt32
   Dim i As Long
   i = c.R
   i = i + 256 * c.G
   i = i + 256 * 256 * c.B
   ToUInt32 = Convert.ToUInt32(i)
End Function
```

The following C# sample changes the background color for the exe files:

```
EXFILEVIEWLib.FileType fileType = axExFileView1.FileTypes.Add("*.exe");
fileType.BackColor = ToUInt32(Color.Red);
fileType.Apply();
```

where the ToUInt32 function converts a Color expression to OLE_COLOR type,

```
private UInt32 ToUInt32(Color c)
{
   long i;
   i = c.R;
   i = i + 256 * c.G;
   i = i + 256 * 256 * c.B;
   return Convert.ToUInt32(i);
}
```

The following VFP sample changes the background color for the exe files:

```
With thisform.ExFileView1.Add("*.exe")
   .BackColor = RGB(255,0,0)
   .Apply()
EndWith
```

method FileTypes.Apply ()

Applies the changes to the current list.

Type	Description

The Apply method applies all the changes to the current list. If the Apply method is not called, the changes will be reflected as soon as the user browses a new folder, or refresh the control. Use the BrowseFolderPath property to specify the browsed folder. The Add method does **not** invoke the Apply method. Use the Refresh method to refresh the control.

The following VB sample makes the cpp and h files appear in italic:

```
With ExFileView1.FileTypes.Add("*.cpp *.h")
   .Italic = True
   .Apply
End With
```

The following C++ sample makes the cpp and h files appear in italic:

```
#include "FileType.h"
#include "FileTypes.h"
CFileType fileType = m_fileview.GetFileTypes().Add("*.cpp *.h");
fileType.SetItalic( TRUE );
fileType.Apply();
```

The following VB.NET sample makes the cpp and h files appear in italic:

```
With AxExFileView1.FileTypes.Add("*.cpp *.h")
   .Italic = True
   .Apply()
End With
```

The following C# sample makes the cpp and h files appear in italic:

```
EXFILEVIEWLib.FileType fileType = axExFileView1.FileTypes.Add("*.cpp *.h");
fileType.Italic = true;
fileType.Apply();
```

The following VFP sample makes the cpp and h files appear in italic:

```
With thisform.ExFileView1.FileTypes.Add("*.cpp *.h")
    .Italic = .t.
    .Apply()
EndWith
```

method FileTypes.Clear ()

Clears all the elements of collection.

Type	Description

Use the Clear method to remove all FileType objects. Use the Apply method to refresh the control's list. Use the Remove method to remove a specified FileType object. The Item property accesses a FileType object by its index. The Count property specifies the number of FileType objects in the control. Use the FileTypes property to access the FileType objects collection.

property FileTypes.Count as Long

Returns the number of objects in a collection.

Type	Description
Long	A long expression that indicates the count of objects into collection.

The Count property counts the element in the collection. Use the Add method to add new FileType objects to the control. Use the Remove method to remove a specified FileType object. Use the Item property to retrieve a specified FileType object. Use the FileTypes property to access the FileType objects collection. Use the Pattern property to specify the group's pattern.

The following VB sample enumerates the FileType objects:

```
With ExFileView1.FileTypes
   Dim i As Long
   For i = 0 To .Count - 1
      Debug.Print .Item(i).Pattern
   Next
End With
```

The following C++ sample enumerates the FileType objects:

```
CFileTypes fileTypes = m_fileview.GetFileTypes();
for ( long i = 0; i < fileTypes.GetCount(); i++ )
   OutputDebugString( fileTypes.GetItem( i ).GetPattern() );
```

The following VB.NET sample enumerates the FileType objects:

```
With AxExFileView1.FileTypes
   Dim i As Integer
   For i = 0 To .Count - 1
      Debug.WriteLine(.Item(i).Pattern())
   Next
End With
```

The following C# sample enumerates the FileType objects:

```
EXFILEVIEWLib.FileTypes files = axExFileView1.FileTypes;
```

```
for (int i = 0; i < files.Count; i++)
   System.Diagnostics.Debug.WriteLine(files[i].Pattern);
```

The following VFP sample enumerates the FileType objects:

```
With thisform.ExFileView1.FileTypes
   local i
   For i = 0 To .Count - 1
      wait window nowait .Item(i).Pattern
   Next
EndWith
```

property FileTypes.Item (Index as Long) as FileType

Retrieves a FileType object given its index into collection.

Type	Description
Index as Long	A long expression that indicates the FileType's index.
FileType	A FileType object being retrieved.

Use the Item property to retrieve a specified FileType object. The Count property counts the element in the collection. Use the Add method to add new FileType objects to the control. Use the Remove method to remove a specified FileType object. Use the FileTypes property to access the FileType objects collection. Use the Pattern property to specify the group's pattern.

The following VB sample enumerates the FileType objects:

```
With ExFileView1.FileTypes
   Dim i As Long
   For i = 0 To .Count - 1
      Debug.Print .Item(i).Pattern
   Next
End With
```

The following C++ sample enumerates the FileType objects:

```
CFileTypes fileTypes = m_fileview.GetFileTypes();
for ( long i = 0; i < fileTypes.GetCount(); i++ )
   OutputDebugString( fileTypes.GetItem( i ).GetPattern() );
```

The following VB.NET sample enumerates the FileType objects:

```
With AxExFileView1.FileTypes
   Dim i As Integer
   For i = 0 To .Count - 1
      Debug.WriteLine(.Item(i).Pattern())
   Next
End With
```

The following C# sample enumerates the FileType objects:

```
EXFILEVIEWLib.FileTypes files = axExFileView1.FileTypes;
for (int i = 0; i < files.Count; i++)
   System.Diagnostics.Debug.WriteLine(files[i].Pattern);
```

The following VFP sample enumerates the FileType objects:

```
With thisform.ExFileView1.FileTypes
   local i
   For i = 0 To .Count - 1
      wait window nowait .Item(i).Pattern
   Next
EndWith
```

method FileTypes.Remove (Index as Long)

Removes a specific member from the FileTypes collection.

Type	Description
Index as Long	A long expression that indicates the FileType's index.

Use the Remove method to remove a specific FileType object. Use Clear method to remove all FileType objects. Use the Apply method to refresh the control's list. The Item property accesses a FileType object by its index. The Count property specifies the number of FileType objects in the control. Use the FileTypes property to access the FileType objects collection.

ExFileView events

The ExFileView control supports the following events:

Name	Description
Change	Fired when the browsed folder changes its content.
Click	Occurs when the user clicks the list.
DblClick	Fired when the user double clicks the item.
Event	Notifies the application once the control fires an event.
FilterChange	Occurs when the filter is changed.
KeyDown	Occurs when the user presses a key while an object has the focus.
KeyPress	Occurs when the user presses and releases an ANSI key.
KeyUp	Occurs when the user releases a key while an object has the focus.
MouseDown	Occur 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.
ScrollButtonClick	Occurs when the user clicks a button in the scrollbar.
Search	Occurs when searching files starts or ends
StateChange	Fired while the control's state has been changed.

event Change (Files as Files)

Fired when the browsed folder has changed its content.

Type	Description
Files as Files	A Files object that contains the list of files that has been changed.

Use the Change event to notify your application whenever the browsed folder's content has been changed, like adding or removing files. The Change event is fired only if the AutoUpdate property is True, and ChangeNotification property is True. Use the State property to determine the new state of the file or folder. Use the Folder property to specify whether the object holds information about a folder of a file. Use the BrowseFolderPath property to indicates the browsed folder. Use the Item property to access a file giving its index in the Files collection. Use the Count property to retrieve the number of File objects in the Files collection.

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

C#

```
private void Change(object sender,exontrol.EXFILEVIEWLib.Files Files)
{
}
```

VB

```
Private Sub Change(ByVal sender As System.Object,ByVal Files As
exontrol.EXFILEVIEWLib.Files) Handles Change
End Sub
```

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

C#

```
private void Change(object sender,
AxEXFILEVIEWLib._IExFileViewEvents_ChangeEvent e)
{
}
```

C++

```
void OnChange(LPDISPATCH Files)
{
}
```

C++ Builder

```
void __fastcall Change(TObject *Sender,Exfileviewlib_tlb::IFiles *Files)
{
}
```

Delphi

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

Delphi 8 (.NET only)

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

Powe…

```
begin event Change(oleobject Files)
end event Change
```

VB.NET

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

VB6

```
Private Sub Change(ByVal Files As EXFILEVIEWLibCtl.IFiles)
End Sub
```

VBA

```
Private Sub Change(ByVal Files As Object)
End Sub
```

VFP

```
LPARAMETERS Files
```

Xbas…

```
PROCEDURE OnChange(oExFileView,Files)
RETURN
```

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

Java…

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

VBSc…

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

Visual Data...

```
Procedure OnComChange Variant llFiles
    Forward Send OnComChange llFiles
End_Procedure
```

Visual Objects

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

X++

```
void onEvent_Change(COM _Files)
{
}
```

XBasic

```
function Change as v (Files as OLE::Exontrol.ExFileView.1::IFiles)
end function
```

dBASE

```
function nativeObject_Change(Files)
return
```

The following VB sample displays the files that have been changed in the browsed folder:

```
Private Sub ExFileView1_Change(ByVal Files As EXFILEVIEWLibCtl.IFiles)
    Dim f As EXFILEVIEWLibCtl.File
    For Each f In Files
        Debug.Print "'" & f.Name & "' " & IIf(f.Folder, "foder", "file") & " " & IIf(f.State = Added,
"added", IIf(f.State = Changed, "changed", IIf(f.State = Deleted, "deleted", "unchanged")))
    Next
End Sub
```

Open a new Windows Explorer instance that browses the same folder as your control. Add new folders, remove folders, or change regular files. Your VB output should look like the following:

```
The 'New Folder (2)' foder - Added
The 'New Folder (2)' foder - Deleted
The 'New Folder' foder was deleted
The 'New Text Document.txt' file was added
The 'New Text Document.txt' file was changed
```

The following C++ sample displays the files that have been changed in the browsed folder:

```
#include "Files.h"
#include "File.h"
void OnChangeExfileview1(LPDISPATCH Files)
{
    CFiles files( Files ); files.m_bAutoRelease = FALSE;
    for ( long i = 0; i < files.GetCount(); i++ )
    {
        CFile1 file = files.GetItem( COleVariant( long( i ) ) );
        CString strState;
        switch ( file.GetState() )
        {
            case 0:
            {
                strState = "unchanged";
                break;
            }
            case 1:
            {
                strState = "changed";
                break;
            }
            case 2:
            {
                strState = "added";
                break;
            }
            case 3:
            {
                strState = "deleted";
                break;
            }
        }
        CString strOutput;
        strOutput.Format( "'%s' %s %s\n", file.GetName(), (file.GetFolder() ? "folder" : "file" ),
strState );
        OutputDebugString( strOutput );
    }
```

```
}
```

The following VB.NET sample displays the files that have been changed in the browsed folder:

```
Private Sub AxExFileView1_Change(ByVal sender As Object, ByVal e As
AxEXFILEVIEWLib._IExFileViewEvents_ChangeEvent) Handles AxExFileView1.Change
   Dim f As EXFILEVIEWLib.File
   For Each f In e.files
      Debug.WriteLine("'" & f.Name & "' " & IIf(f.Folder, "foder", "file") & " " & IIf(f.State =
EXFILEVIEWLib.ChangeEnum.Added, "added", IIf(f.State =
EXFILEVIEWLib.ChangeEnum.Changed, "changed", IIf(f.State =
EXFILEVIEWLib.ChangeEnum.Deleted, "deleted", "unchanged"))))
   Next
End Sub
```

The following C# sample displays the files that have been changed in the browsed folder:

```
private void axExFileView1_Change(object sender,
AxEXFILEVIEWLib._IExFileViewEvents_ChangeEvent e)
{
   for (int i = 0; i < e.files.Count; i++)
   {
      EXFILEVIEWLib.File file = e.files[i];
      string strOutput = "'" + file.Name + "' ";
      strOutput += (file.Folder ? "folder" : "file") + " ";
      strOutput += (file.State == EXFILEVIEWLib.ChangeEnum.Added ? "added" : (file.State
== EXFILEVIEWLib.ChangeEnum.Deleted ? "deleted" : (file.State ==
EXFILEVIEWLib.ChangeEnum.Changed ? "changed" : "unchanged")));
      System.Diagnostics.Debug.WriteLine(strOutput);
   }
}
```

The following VFP sample displays the files that have been changed in the browsed folder:

```
*** ActiveX Control Event ***
LPARAMETERS files

with files
```

```
    local i
    for i = 0 to .Count - 1
        with .Item(i)
            wait window nowait .Name + " " + str(.State)
        endwith
    next
endwith
```

event Click ()

Occurs when the user clicks the list.

Type	Description

Use the Click event to notify your application when the user clicks the list. Use Get property to retrieve the collection of selected items. Use the StateChange event to notify your application when the current selection is changed. Use the FileFromPoint property to retrieve the file from the cursor. Use the MouseDown or MouseUp event to notify your application when the user presses or releases the one of the mouse buttons.

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

C#

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

VB

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

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

C#

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

C++

```
void OnClick()
{
}
```

C++ Builder

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

Delphi

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

Delphi 8 (.NET only)

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

Powe…

```
begin event Click()
end event Click
```

VB.NET

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

VB6

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

VBA

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

VFP

```
LPARAMETERS nop
```

Xbas…

```
PROCEDURE OnClick(oExFileView)
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
```

The following VB sample prints the selected file or folder, when the user clicks the control's list:

```
Private Sub ExFileView1_Click()
   Dim fs As Files, f As File
   Set fs = ExFileView1.Get(SelItems)
   For Each f In fs
      Debug.Print f.Name
   Next
End Sub
```

The following C++ sample prints the selected file or folder, when the user clicks the control's list:

```
void OnClickExfileview1()
{
   CFiles files = m_fileview.GetGet( 0 /*SelItems*/ );
   for ( long i = 0; i < files.GetCount(); i++ )
   {
      CFile1 file = files.GetItem( COleVariant( i ) );
      OutputDebugString( file.GetName() );
   }
}
```

The following VB.NET sample prints the selected file or folder, when the user clicks the control's list:

```
Private Sub AxExFileView1_ClickEvent(ByVal sender As Object, ByVal e As
System.EventArgs) Handles AxExFileView1.ClickEvent
   Dim f As EXFILEVIEWLib.File
   For Each f In AxExFileView1.get_Get(EXFILEVIEWLib.TypeEnum.SelItems)
      Debug.WriteLine(f.Name)
   Next
End Sub
```

The following C# sample prints the selected file or folder, when the user clicks the control's list:

```
private void axExFileView1_ClickEvent(object sender, EventArgs e)
{
   EXFILEVIEWLib.Files files = axExFileView1.get_Get(EXFILEVIEWLib.TypeEnum.SelItems);
   for (int i = 0; i < files.Count; i++)
   {
      EXFILEVIEWLib.File file = files[i];
      System.Diagnostics.Debug.WriteLine(file.Name);
   }
}
```

The following VFP sample prints the selected file or folder, when the user clicks the control's list:

```
*** ActiveX Control Event ***

with thisform.ExFileView1.Get(0)
   local i
   for i = 0 to .Count - 1
      with .Item(i)
         wait window nowait .Name
      endwith
   next
endwith
```

event DblClick ()

Fired when the user double clicks an item.

Type	Description

The DblClk event is fired whenever the user double clicks a file or a folder. Use the ExpandOnDblClk property to specify whether the folder is expanded or collapsed when the user double clicks it. By default, if the user double clicks a folder , the control browses for a new folder. Use the BrowseFolderPath property to specify the path to the browsed folder. Use StateChange event to notify your application when the current selection is changed. Use the Get property to retrieve the selected item(s). Use the Folder property to specify whether the File object holds a file or a folder. Use the ExecuteContextCommand method to invoke a command from the file's context menu.

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

C#

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

VB

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

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

C#

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

C++

```
void OnDblClick()
{
}
```

C++ Builder

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

Delphi

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

Delphi 8 (.NET only)

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

Powe…

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

VB.NET

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

VB6

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

VBA

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

VFP

```
LPARAMETERS nop
```

Xbas…

```
PROCEDURE OnDblClick(oExFileView)
RETURN
```

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

Java…

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

VBSc…

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

Visual Data…

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

Visual Objects

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

X++

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

XBasic

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

dBASE

```
function nativeObject_DblClick()
return
```

The following VB sample opens that file being double clicked:

```
Private Sub ExFileView1_DblClick()
   With ExFileView1.Get(SelItems)
      If (.Count > 0) Then
         With .Item(0)
            If (Not .Folder) Then
               ExFileView1.ExecuteContextCommand .Name, .Folder, "Open"
            End If
         End With
      End If
   End With
End Sub
```

The following C++ sample opens that file being double clicked:

```
void OnDblClickExfileview1()
{
   CFiles files = m_fileview.GetGet( 0 );
   if ( files.GetCount() > 0 )
   {
      CFile1 file = files.GetItem( COleVariant( (long)0 ) );
      m_fileview.ExecuteContextCommand( file.GetName(), file.GetFolder(), "Open" );
```

```
    }
}
```

The following VB.NET sample opens that file being double clicked:

```
Private Sub AxExFileView1_DblClick(ByVal sender As Object, ByVal e As System.EventArgs)
Handles AxExFileView1.DblClick
    With AxExFileView1.get_Get(EXFILEVIEWLib.TypeEnum.SelItems)
        If (.Count > 0) Then
            With .Item(0)
                If (Not .Folder) Then
                    AxExFileView1.ExecuteContextCommand(.Name, .Folder, "Open")
                End If
            End With
        End If
    End With
End Sub
```

The following C# sample opens that file being double clicked:

```
private void axExFileView1_DblClick(object sender, EventArgs e)
{
    EXFILEVIEWLib.Files files = axExFileView1.get_Get(EXFILEVIEWLib.TypeEnum.SelItems);
    if (files.Count > 0)
    {
        EXFILEVIEWLib.File file = files[0];
        axExFileView1.ExecuteContextCommand(file.Name, file.Folder, "Open");
    }
}
```

The following VFP sample opens that file being double clicked:

```
*** ActiveX Control Event ***

With thisform.ExFileView1.Get(0)    && SelItems
    If (.Count > 0) Then
        With .Item(0)
            If (Not .Folder) Then
                thisform.ExFileView1.ExecuteContextCommand(.Name, .Folder, "Open")
```

```
        EndIf
      EndWith
    EndIf
EndWith
```

event Event (EventID as Long)

Notifies the application once the control fires an event.

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

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

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

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

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

C#

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

VB

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

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

C#

```
private void Event(object sender, AxEXFILEVIEWLib._IExFileViewEvents_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:
AxEXFILEVIEWLib._IExFileViewEvents_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
AxEXFILEVIEWLib._IExFileViewEvents_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(oExFileView,EventID)
RETURN
```

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

Java…

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

VBSc…

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

Visual Data…

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

Visual Objects

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

X++

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

XBasic

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

dBASE

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

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

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

```
// Notifies the application once the control fires an event.
void onEvent_Event(int _EventID)
{
   print exfileview1.EventParam(-2).toString();
}
```

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

in the list bellow:

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

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

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

```
// Notifies the application once the control fires an event.
void onEvent_Event(int _EventID)
{
   ;
   if ( _EventID == 125 ) /*event BarParentChange (Item as HITEM, Key as Variant, NewItem as HITEM, Cancel as Boolean) */
      exfileview1.EventParam( 3 /*Cancel*/, COMVariant::createFromBoolean(true) );
}
```

The code checks if the BarParentChange (_EventID == 125) event is fired, and changes the third parameter of the event to true. The definition for BarParentChange event can be consulted in the control's documentation or in the ActiveX explorer. So, anytime you need to access the original parameters for the event you should use the EventParam method that allows you to get or set a parameter. If the parameter is not passed by reference, you can not change the parameter's value.

Now, let's add some code to see a complex sample, so let's say that we need to prevent moving the bar from an item to any disabled item. So, we need to specify the Cancel parameter as not Items.EnableItem(NewItem), in other words cancels if the new parent is disabled. Shortly the code will be:

```
// Notifies the application once the control fires an event.
void onEvent_Event(int _EventID)
{
   ;
```

```
    if ( _EventID == 125 ) /*event BarParentChange (Item as HITEM, Key as Variant, NewItem as HITEM, Cancel as Boolean) */
        if ( !exfileview1.Items().EnableItem( exfileview1.EventParam( 2 /*NewItem*/ ) ) )
            exfileview1.EventParam( 3 /*Cancel*/, COMVariant::createFromBoolean(true) );
}
```

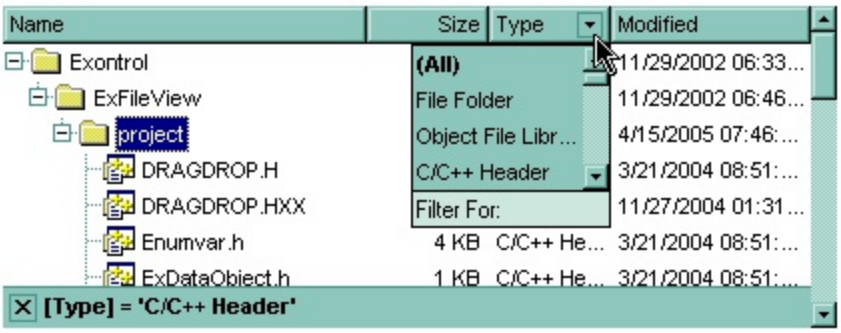
In conclusion, anytime the X++ fires the "invalid parameters." while handling an event, you can use and handle the Event notification and EventParam methods of the control

event FilterChange ()

Occurs when the filter is changed. /*not supported in the lite version*/

Type	Description

The FilterChange event notifies your application that the user filters files/folders in the control. Use the ColumnFilterButton property to display a filter button in the column's caption. Use the AddColumnCustomFilter method to add custom filter patterns to the column. Use the ColumnFilter, ColumnFilterType properties and ApplyFilter method to apply a filter to the control's content. Use the Get property to retrieve all, selected or checked items. Use the Folder property to specify whether the File object holds a file or a folder.



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

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

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

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

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

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

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

```
}
```

Delphi

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

Delphi 8 (.NET only)

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

Powe…

```
begin event FilterChange()
end event FilterChange
```

VB.NET

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

VB6

```
Private Sub FilterChange()
End Sub
```

VBA

```
Private Sub FilterChange()
End Sub
```

VFP

```
LPARAMETERS nop
```

Xbas…

```
PROCEDURE OnFilterChange(oExFileView)
RETURN
```

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

Java…

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

VBSc…

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

Visual Data…

```
Procedure OnComFilterChange
    Forward Send OnComFilterChange
End_Procedure
```

Visual Objects

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

X++

```
void onEvent_FilterChange()
{
}
```

XBasic

```
function FilterChange as v ()
end function
```

dBASE

```
function nativeObject_FilterChange()
return
```

The following VB sample displays the list of files once that user changes the filter:

```
Private Sub ExFileView1_FilterChange()
    With ExFileView1.Get(VisibleItems)
        For i = 0 To .Count - 1
            With .Item(i)
                Debug.Print .Name
            End With
        Next
    End With
End Sub
```

The following C++ sample displays the list of files once that user changes the filter:

```
void OnFilterChangeExfileview1()
{
    CFiles files = m_fileview.GetGet( 3 /*VisibleItems*/ );
    for ( long i = 0; i < files.GetCount(); i++ )
        OutputDebugString( files.GetItem( COleVariant( i ) ).GetName() );
}
```

The following VB.NET sample displays the list of files once that user changes the filter:

```
Private Sub AxExFileView1_FilterChange(ByVal sender As Object, ByVal e As
System.EventArgs) Handles AxExFileView1.FilterChange
   With AxExFileView1.get_Get(EXFILEVIEWLib.TypeEnum.VisibleItems)
      Dim i As Integer
      For i = 0 To .Count - 1
         With .Item(i)
            Debug.WriteLine(.Name())
         End With
      Next
   End With
End Sub
```

The following C# sample displays the list of files once that user changes the filter:

```
private void axExFileView1_FilterChange(object sender, EventArgs e)
{
   EXFILEVIEWLib.Files files =
axExFileView1.get_Get(EXFILEVIEWLib.TypeEnum.VisibleItems);
   for (int i = 0; i < files.Count; i++)
   {
      EXFILEVIEWLib.File file = files[i];
      System.Diagnostics.Debug.WriteLine(file.Name);
   }
}
```

The following VFP sample displays the list of files once that user changes the filter:

```
*** ActiveX Control Event ***

With thisform.ExFileView1.Get(3)    && VisibleItems
   For i = 0 To .Count - 1
      With .Item(i)
         wait window nowait .Name
      EndWith
   Next
EndWith
```

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. The control fires the StateChange when the user selects a new file or folder. 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,
AxEXFILEVIEWLib._IExFileViewEvents_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:
AxEXFILEVIEWLib._IExFileViewEvents_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
AxEXFILEVIEWLib._IExFileViewEvents_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(oExFileView,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,
AxEXFILEVIEWLib._IExFileViewEvents_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:
AxEXFILEVIEWLib._IExFileViewEvents_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
AxEXFILEVIEWLib._IExFileViewEvents_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(oExFileView,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)

Occur 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,
AxEXFILEVIEWLib._IExFileViewEvents_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:
AxEXFILEVIEWLib._IExFileViewEvents_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
AxEXFILEVIEWLib._IExFileViewEvents_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(oExFileView,KeyCode,Shift)
RETURN
```

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

Java…

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

VBSc…

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

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

Visual Data…

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

Visual Objects

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

X++

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

XBasic

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

dBASE

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

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

Occur when the user presses a mouse button.

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

Use 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 FileFromPoint property to retrieve the file from the cursor. The control fires the StateChange event when the selection is changed. Use Get property to retrieve the collection of selected items.

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,
AxEXFILEVIEWLib._IExFileViewEvents_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:
AxEXFILEVIEWLib._IExFileViewEvents_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
AxEXFILEVIEWLib._IExFileViewEvents_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(oExFileView,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.ExFileView.1::OLE_XPOS_PIXELS,Y as
OLE::Exontrol.ExFileView.1::OLE_YPOS_PIXELS)
end function
```

dBASE

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

The following VB sample displays the file or the folder being clicked:

```
Private Sub ExFileView1_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As
Single)
```

```
    Dim f As String
    f = ExFileView1.FileFromPoint(X / Screen.TwipsPerPixelX, Y / Screen.TwipsPerPixelY)
    If Len(f) > 0 Then
        Debug.Print f
    End If
End Sub
```

The following C++ sample displays the file or the folder being clicked:

```
void OnMouseDownExfileview1(short Button, short Shift, long X, long Y)
{
    CString f = m_fileview.GetFileFromPoint( X, Y );
    if ( f.GetLength() > 0 )
        OutputDebugString( f );
}
```

The following VB.NET sample displays the file or the folder being clicked:

```
Private Sub AxExFileView1_MouseDownEvent(ByVal sender As Object, ByVal e As
AxEXFILEVIEWLib._IExFileViewEvents_MouseDownEvent) Handles
AxExFileView1.MouseDownEvent
    Dim f As String = AxExFileView1.get_FileFromPoint(e.x, e.y)
    If Len(f) > 0 Then
        Debug.WriteLine(f)
    End If
End Sub
```

The following C# sample displays the file or the folder being clicked:

```
private void axExFileView1_MouseDownEvent(object sender,
AxEXFILEVIEWLib._IExFileViewEvents_MouseDownEvent e)
{
    string f = axExFileView1.get_FileFromPoint(e.x, e.y);
    if (f.Length > 0)
        System.Diagnostics.Debug.WriteLine(f);
}
```

The following VFP sample displays the file or the folder being clicked:

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

with thisform.ExFileView1
   local f
   f = .FileFromPoint( x, y )
   if ( len(f) > 0 )
      wait window nowait f
   endif
endwith
```

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.

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,
AxEXFILEVIEWLib._IExFileViewEvents_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:
AxEXFILEVIEWLib._IExFileViewEvents_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
AxEXFILEVIEWLib._IExFileViewEvents_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(oExFileView,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.ExFileView.1::OLE_XPOS_PIXELS,Y as
OLE::Exontrol.ExFileView.1::OLE_YPOS_PIXELS)
end function
```

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

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 FileFromPoint property to retrieve the file from the cursor. The control fires the StateChange event when the selection is changed. Use Get property to retrieve the collection of selected items.

The following VB sample displays the file from the cursor:

```
Private Sub ExFileView1_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As
```

```
Single)
   Dim f As String
   f = ExFileView1.FileFromPoint(X / Screen.TwipsPerPixelX, Y / Screen.TwipsPerPixelY)
   If Len(f) > 0 Then
      Debug.Print f
   End If
End Sub
```

The following C++ sample displays the file from the cursor:

```
void OnMouseMoveExfileview1(short Button, short Shift, long X, long Y)
{
   CString f = m_fileview.GetFileFromPoint( X, Y );
   if ( f.GetLength() > 0 )
      OutputDebugString( f );
}
```

The following VB.NET sample displays the file from the cursor:

```
Private Sub AxExFileView1_MouseMoveEvent(ByVal sender As Object, ByVal e As
AxEXFILEVIEWLib._IExFileViewEvents_MouseMoveEvent) Handles
AxExFileView1.MouseMoveEvent
   Dim f As String = AxExFileView1.get_FileFromPoint(e.x, e.y)
   If Len(f) > 0 Then
      Debug.WriteLine(f)
   End If
End Sub
```

The following C# sample displays the file from the cursor:

```
private void axExFileView1_MouseMoveEvent(object sender,
AxEXFILEVIEWLib._IExFileViewEvents_MouseMoveEvent e)
{
   string f = axExFileView1.get_FileFromPoint(e.x, e.y);
   if (f.Length > 0)
      System.Diagnostics.Debug.WriteLine(f);
}
```

The following VFP sample displays the file from the cursor:

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

with thisform.ExFileView1
   local f
   f = .FileFromPoint( x, y )
   if ( len(f) > 0 )
      wait window nowait f
   endif
endwith
```

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. the FileFromPoint property to retrieve the file from the cursor. The control fires the StateChange event when the selection is changed. Use Get property to retrieve the collection of selected items.

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,
AxEXFILEVIEWLib._IExFileViewEvents_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:
AxEXFILEVIEWLib._IExFileViewEvents_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
AxEXFILEVIEWLib._IExFileViewEvents_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(oExFileView,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.ExFileView.1::OLE_XPOS_PIXELS,Y as
OLE::Exontrol.ExFileView.1::OLE_YPOS_PIXELS)
end function
```

dBASE

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

The following VB sample displays the file or the folder being clicked:

```
Private Sub ExFileView1_MouseUp(Button As Integer, Shift As Integer, X As Single, Y As
```

```
Single)
   Dim f As String
   f = ExFileView1.FileFromPoint(X / Screen.TwipsPerPixelX, Y / Screen.TwipsPerPixelY)
   If Len(f) > 0 Then
      Debug.Print f
   End If
End Sub
```

The following C++ sample displays the file or the folder being clicked:

```
void OnMouseUpExfileview1(short Button, short Shift, long X, long Y)
{
   CString f = m_fileview.GetFileFromPoint( X, Y );
   if ( f.GetLength() > 0 )
      OutputDebugString( f );
}
```

The following VB.NET sample displays the file or the folder being clicked:

```
Private Sub AxExFileView1_MouseUpEvent(ByVal sender As Object, ByVal e As
AxEXFILEVIEWLib._IExFileViewEvents_MouseUpEvent) Handles
AxExFileView1.MouseUpEvent
   Dim f As String = AxExFileView1.get_FileFromPoint(e.x, e.y)
   If Len(f) > 0 Then
      Debug.WriteLine(f)
   End If
End Sub
```

The following C# sample displays the file or the folder being clicked:

```
private void axExFileView1_MouseUpEvent(object sender,
AxEXFILEVIEWLib._IExFileViewEvents_MouseUpEvent e)
{
   string f = axExFileView1.get_FileFromPoint(e.x, e.y);
   if (f.Length > 0)
      System.Diagnostics.Debug.WriteLine(f);
}
```

The following VFP sample displays the file or the folder being clicked:

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

with thisform.ExFileView1
   local f
   f = .FileFromPoint( x, y )
   if ( len(f) > 0 )
      wait window nowait f
   endif
endwith
```

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 ExFileView control only supports manual OLE drag and drop events. In order to enable OLE drag and drop feature into ExFileView control you have to check 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,
```

```
AxEXFILEVIEWLib._IExFileViewEvents_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:
AxEXFILEVIEWLib._IExFileViewEvents_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
AxEXFILEVIEWLib._IExFileViewEvents_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(oExFileView,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.

The OLEDragDrop event is fired when the user has dropped files or clipboard information into ExFileView control. Use the OLEDropMode property on exOLEDropManual to enable OLE drag drop support. Use the FileFromPoint property to retrieve the file from the cursor.

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, AxEXFILEVIEWLib._IExFileViewEvents_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,Exfileviewlib_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:
AxEXFILEVIEWLib._IExFileViewEvents_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
AxEXFILEVIEWLib._IExFileViewEvents_OLEDragDropEvent) Handles OLEDragDrop
End Sub
```

VB6

```
Private Sub OLEDragDrop(ByVal Data As EXFILEVIEWLibCtl.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(oExFileView,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.ExFileView.1::IExDataObject,Effect as N,Button as N,Shift as N,X as
OLE::Exontrol.ExFileView.1::OLE_XPOS_PIXELS,Y as
OLE::Exontrol.ExFileView.1::OLE_YPOS_PIXELS)
end function
```

dBASE

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

The following VB sample displays the list of files being dragged to the control (open your Windows Explorer, select some files and drag them to the control) :

```
Private Sub ExFileView1_OLEDragDrop(ByVal Data As EXFILEVIEWLibCtl.IExDataObject,
Effect As Long, ByVal Button As Integer, ByVal Shift As Integer, ByVal X As Single, ByVal Y As
Single)
   With Data.Files
      Dim i As Long
      For i = 0 To .Count - 1
         Debug.Print .Item(i)
      Next
   End With
End Sub
```

The following C++ sample displays the list of files being dragged to the control:

```
#import <exfilevw.dll>
void OnOLEDragDropExfileview1(LPDISPATCH Data, long FAR* Effect, short Button, short
Shift, long X, long Y)
{
   EXFILEVIEWLib::IExDataObjectPtr spData( Data );
   if ( spData )
   {
      EXFILEVIEWLib::IExDataObjectFilesPtr spFiles = spData->Files;
      for ( long i = 0; i < spFiles->Count; i++ )
         OutputDebugString( spFiles->Item[ i ] );
   }
}
```

The C++ requires #import <exfilevw.dll> to import definitions for ExDataObject and ExDataObjectFiles objects. The #import <exfilevw.dll> generates the EXFILEVIEWLib namespace. If the exfilevw.dll file is located in other directory than system folder, the correct path should be provided, else a compiler error occurs.

The following VB.NET sample displays the list of files being dragged to the control:

```
Private Sub AxExFileView1_OLEDragDrop(ByVal sender As Object, ByVal e As
AxEXFILEVIEWLib._IExFileViewEvents_OLEDragDropEvent) Handles
AxExFileView1.OLEDragDrop
   With e.data.Files
      Dim i As Long
      For i = 0 To .Count - 1
         Debug.WriteLine(.Item(i))
      Next
   End With
End Sub
```

The following C# sample displays the list of files being dragged to the control:

```
private void axExFileView1_OLEDragDrop(object sender,
AxEXFILEVIEWLib._IExFileViewEvents_OLEDragDropEvent e)
{
   EXFILEVIEWLib.ExDataObjectFiles files = e.data.Files;
```

```
    for (int i = 0; i < files.Count; i++)
        System.Diagnostics.Debug.WriteLine(files[i]);
}
```

The following VFP sample displays the list of files being dragged to the control:

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

With data.Files
    local i
    For i = 0 To .Count - 1
        wait window nowait .Item(i)
    Next
EndWith
```

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.

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, AxEXFILEVIEWLib._IExFileViewEvents_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,Exfileviewlib_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:
AxEXFILEVIEWLib._IExFileViewEvents_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
AxEXFILEVIEWLib._IExFileViewEvents_OLEDragOverEvent) Handles OLEDragOver
End Sub
```

VB6

```
Private Sub OLEDragOver(ByVal Data As EXFILEVIEWLibCtl.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(oExFileView,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.ExFileView.1::IExDataObject,Effect as N,Button as N,Shift as N,X as
OLE::Exontrol.ExFileView.1::OLE_XPOS_PIXELS,Y as
OLE::Exontrol.ExFileView.1::OLE_YPOS_PIXELS,State as N)
end function
```

dBASE

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

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 ExFileView control only supports manual OLE drag and drop events.

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 ExFileView control only supports 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,
AxEXFILEVIEWLib._IExFileViewEvents_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:
AxEXFILEVIEWLib._IExFileViewEvents_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
AxEXFILEVIEWLib._IExFileViewEvents_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(oExFileView,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 implemented.

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, AxEXFILEVIEWLib._IExFileViewEvents_OLESetDataEvent e)
{
}
```

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

C++ Builder
```
void __fastcall OLESetData(TObject *Sender,Exfileviewlib_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:
AxEXFILEVIEWLib._IExFileViewEvents_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
AxEXFILEVIEWLib._IExFileViewEvents_OLESetDataEvent) Handles OLESetData
End Sub
```

VB6

```
Private Sub OLESetData(ByVal Data As EXFILEVIEWLibCtl.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(oExFileView,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.ExFileView.1::IExDataObject,Format as N)
end function
```

dBASE

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

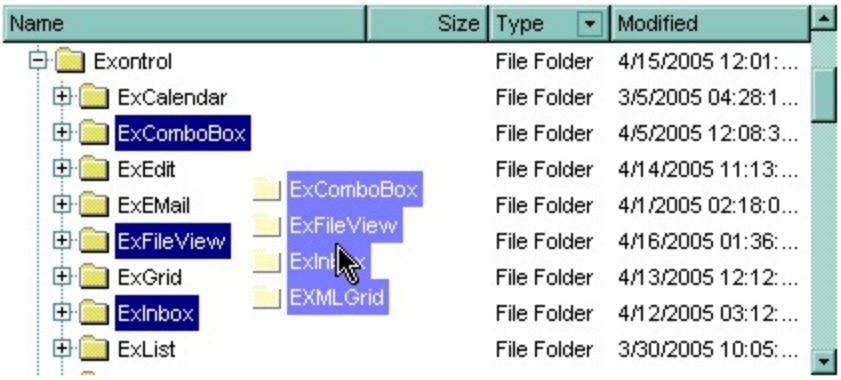
event OLEStartDrag (Data as ExDataObject, AllowedEffects as Long)

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.

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.



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,
AxEXFILEVIEWLib._IExFileViewEvents_OLEStartDragEvent e)
{
}
```

C++
```
void OnOLEStartDrag(LPDISPATCH Data,long FAR* AllowedEffects)
{
}
```

C++ Builder
```
void __fastcall OLEStartDrag(TObject *Sender,Exfileviewlib_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:
AxEXFILEVIEWLib._IExFileViewEvents_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
AxEXFILEVIEWLib._IExFileViewEvents_OLEStartDragEvent) Handles OLEStartDrag
End Sub
```

VB6
```
Private Sub OLEStartDrag(ByVal Data As
EXFILEVIEWLibCtl.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(oExFileView,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.ExFileView.1::IExDataObject,AllowedEffects as N)
end function
```

dBASE

```
function nativeObject_OLEStartDrag(Data,AllowedEffects)
return
```

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 the Get property to retrieve the selected items. Use the FullName property to retrieve the full name of the file. You can use the RegisterClipboardFormat API function to register a new clipboard format. This format can then be used as a valid clipboard format. Use the SingleSel property to allow multiple selection in the control. The control fires the OLEDragDrop event when the user drags data over the control.

The user can drag files from the component only if:

- OLEDropMode property is exOLEDropManual
- The AllowedEffects parameter in the OLEStartDrag event is not zero
- The Files or the SetData method is called during the OLEStartDrag event

The following VB sample starts dragging the selected files:

```
Private Sub ExFileView1_OLEStartDrag(ByVal Data As ExDataObject, AllowedEffects As
Long)
   Data.Files.Clear
   With ExFileView1.Get(SelItems)
      Dim i As Long
      For i = 0 To .Count - 1
       Data.Files.Add .Item(i).FullName
      Next
   End With
   If (Data.Files.Count > 0) Then
      AllowedEffects = 1
      Data.SetData , exCFFiles
   End If
End Sub
```

The following C++ sample starts dragging the selected files:

```
#import <exfilevw.dll>
void OnOLEStartDragExfileview1(LPDISPATCH Data, long FAR* AllowedEffects)
{
   EXFILEVIEWLib::IExDataObjectPtr spData( Data );
   spData->Clear();
   CFiles files = m_fileview.GetGet( 0 /*SelItems*/ );
```

```
    for ( long i = 0; i < files.GetCount(); i++ )
        spData->Files->Add( files.GetItem( COleVariant( i ) ).GetFullName().operator
LPCTSTR() );
    if ( spData->Files->Count > 0 )
    {
        *AllowedEffects = 1; /*exOLEDropEffectCopy*/
        spData->SetData( vtMissing, COleVariant( long(15) ) ); /*exCFFiles*/
    }
}
```

The C++ requires #import <exfilevw.dll> to import definitions for ExDataObject and ExDataObjectFiles objects. The #import <exfilevw.dll> generates the EXFILEVIEWLib namespace. If the exfilevw.dll file is located in other directory than system folder, the correct path should be provided, else a compiler error occurs.

The following VB.NET sample starts dragging the selected files:

```
Private Sub AxExFileView1_OLEStartDrag(ByVal sender As Object, ByVal e As
AxEXFILEVIEWLib._IExFileViewEvents_OLEStartDragEvent) Handles
AxExFileView1.OLEStartDrag
    e.data.Files.Clear()
    With AxExFileView1.get_Get(EXFILEVIEWLib.TypeEnum.SelItems)
        Dim i As Integer
        For i = 0 To .Count - 1
            e.data.Files.Add(.Item(i).FullName())
        Next
    End With
    If (e.data.Files.Count > 0) Then
        e.allowedEffects = 1
        e.data.SetData(, EXFILEVIEWLib.exClipboardFormatEnum.exCFFiles)
    End If
End Sub
```

The following C# sample starts dragging the selected files:

```
private void axExFileView1_OLEStartDrag(object sender,
AxEXFILEVIEWLib._IExFileViewEvents_OLEStartDragEvent e)
{
    e.data.Files.Clear();
```

```
    EXFILEVIEWLib.Files files = axExFileView1.get_Get(EXFILEVIEWLib.TypeEnum.SelItems);
    for ( int i = 0 ; i < files.Count; i++ )
        e.data.Files.Add(files[i].FullName);
    if (e.data.Files.Count > 0)
    {
        e.allowedEffects = 1;
        e.data.SetData(null, EXFILEVIEWLib.exClipboardFormatEnum.exCFFiles);
    }
}
```

The following VFP sample starts dragging the selected files:

```
*** ActiveX Control Event ***
LPARAMETERS data, allowedeffects

Data.Files.Clear
With thisform.ExFileView1.Get(0) && SelItems
    local i
    For i = 0 To .Count - 1
     data.Files.Add(.Item(i).FullName)
    Next
EndWith
If (Data.Files.Count > 0) Then
    AllowedEffects = 1
    data.SetData( , 15) && exCFFiles
EndIf
```

event ScrollButtonClick (ScrollBar as ScrollBarEnum, ScrollPart as ScrollPartEnum)

Occurs when the user clicks a button in the scrollbar.

Type	Description
ScrollBar as ScrollBarEnum	A ScrollBarEnum expression that specifies the scrollbar being clicked.
ScrollPart as ScrollPartEnum	A ScrollPartEnum expression that indicates the part of the scroll being clicked.

Use the ScrollButtonClick event to notify your application that the user clicks a button in the control's scrollbar. The ScrollButtonClick event is fired when the user clicks and releases the mouse over an enabled part of the scroll bar. Use the 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 Background property to change the visual appearance for any part in the control's scroll bar.

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

C#

```
private void ScrollButtonClick(object
sender,exontrol.EXFILEVIEWLib.ScrollBarEnum
ScrollBar,exontrol.EXFILEVIEWLib.ScrollPartEnum ScrollPart)
{
}
```

VB

```
Private Sub ScrollButtonClick(ByVal sender As System.Object,ByVal ScrollBar As
exontrol.EXFILEVIEWLib.ScrollBarEnum,ByVal ScrollPart As
exontrol.EXFILEVIEWLib.ScrollPartEnum) Handles ScrollButtonClick
End Sub
```

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

C#

```
private void ScrollButtonClick(object sender,
AxEXFILEVIEWLib._IExFileViewEvents_ScrollButtonClickEvent e)
{
}
```

C++

```
void OnScrollButtonClick(long ScrollBar,long ScrollPart)
{
```

```
}
```

C++ Builder

```
void __fastcall ScrollButtonClick(TObject *Sender,Exfileviewlib_tlb::ScrollBarEnum
ScrollBar,Exfileviewlib_tlb::ScrollPartEnum ScrollPart)
{
}
```

Delphi

```
procedure ScrollButtonClick(ASender: TObject; ScrollBar :
ScrollBarEnum;ScrollPart : ScrollPartEnum);
begin
end;
```

Delphi 8 (.NET only)

```
procedure ScrollButtonClick(sender: System.Object; e:
AxEXFILEVIEWLib._IExFileViewEvents_ScrollButtonClickEvent);
begin
end;
```

Powe…

```
begin event ScrollButtonClick(long ScrollBar,long ScrollPart)
end event ScrollButtonClick
```

VB.NET

```
Private Sub ScrollButtonClick(ByVal sender As System.Object, ByVal e As
AxEXFILEVIEWLib._IExFileViewEvents_ScrollButtonClickEvent) Handles
ScrollButtonClick
End Sub
```

VB6

```
Private Sub ScrollButtonClick(ByVal ScrollBar As
EXFILEVIEWLibCtl.ScrollBarEnum,ByVal ScrollPart As
EXFILEVIEWLibCtl.ScrollPartEnum)
End Sub
```

VBA

```
Private Sub ScrollButtonClick(ByVal ScrollBar As Long,ByVal ScrollPart As Long)
End Sub
```

VFP

```
LPARAMETERS ScrollBar,ScrollPart
```

Xbas…

```
PROCEDURE OnScrollButtonClick(oExFileView,ScrollBar,ScrollPart)
RETURN
```

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

Java…
```
<SCRIPT EVENT="ScrollButtonClick(ScrollBar,ScrollPart)" LANGUAGE="JScript">
</SCRIPT>
```

VBSc…
```
<SCRIPT LANGUAGE="VBScript">
Function ScrollButtonClick(ScrollBar,ScrollPart)
End Function
</SCRIPT>
```

Visual Data…
```
Procedure OnComScrollButtonClick OLEScrollBarEnum llScrollBar
OLEScrollPartEnum llScrollPart
   Forward Send OnComScrollButtonClick llScrollBar llScrollPart
End_Procedure
```

Visual Objects
```
METHOD OCX_ScrollButtonClick(ScrollBar,ScrollPart) CLASS MainDialog
RETURN NIL
```

X++
```
void onEvent_ScrollButtonClick(int _ScrollBar,int _ScrollPart)
{
}
```

XBasic
```
function ScrollButtonClick as v (ScrollBar as
OLE::Exontrol.ExFileView.1::ScrollBarEnum,ScrollPart as
OLE::Exontrol.ExFileView.1::ScrollPartEnum)
end function
```

dBASE
```
function nativeObject_ScrollButtonClick(ScrollBar,ScrollPart)
return
```

The following VB sample displays the identifier of the scroll's button being clicked:

```
With ExFileView1
   .BeginUpdate
      .ScrollPartVisible(exVScroll, exLeftB1Part Or exRightB1Part) = True
      .ScrollPartCaption(exVScroll, exLeftB1Part) = "<img></img>1"
      .ScrollPartCaption(exVScroll, exRightB1Part) = "<img></img>2"
```

```
   .EndUpdate
End With
```

```
Private Sub ExFileView1_ScrollButtonClick(ByVal ScrollPart As
EXEXFILEVIEWLibCtl.ScrollPartEnum)
   MsgBox (ScrollPart)
End Sub
```

The following VB.NET sample displays the identifier of the scroll's button being clicked:

```
With AxExFileView1
   .BeginUpdate()
   .set_ScrollPartVisible(EXEXFILEVIEWLib.ScrollBarEnum.exVScroll,
EXEXFILEVIEWLib.ScrollPartEnum.exLeftB1Part Or
EXEXFILEVIEWLib.ScrollPartEnum.exRightB1Part, True)
   .set_ScrollPartCaption(EXEXFILEVIEWLib.ScrollBarEnum.exVScroll,
EXEXFILEVIEWLib.ScrollPartEnum.exLeftB1Part, "<img></img>1")
   .set_ScrollPartCaption(EXEXFILEVIEWLib.ScrollBarEnum.exVScroll,
EXEXFILEVIEWLib.ScrollPartEnum.exRightB1Part, "<img></img>2")
   .EndUpdate()
End With
```

```
Private Sub AxExFileView1_ScrollButtonClick(ByVal sender As System.Object, ByVal e As
AxEXEXFILEVIEWLib._IExFileViewEvents_ScrollButtonClickEvent) Handles
AxExFileView1.ScrollButtonClick
      MessageBox.Show( e.scrollPart.ToString())
End Sub
```

The following C# sample displays the identifier of the scroll's button being clicked:

```
axExFileView1.BeginUpdate();
axExFileView1.set_ScrollPartVisible(EXEXFILEVIEWLib.ScrollBarEnum.exVScroll,
EXEXFILEVIEWLib.ScrollPartEnum.exLeftB1Part |
EXEXFILEVIEWLib.ScrollPartEnum.exRightB1Part, true);
axExFileView1.set_ScrollPartCaption(EXEXFILEVIEWLib.ScrollBarEnum.exVScroll,
EXEXFILEVIEWLib.ScrollPartEnum.exLeftB1Part , "<img></img>1");
axExFileView1.set_ScrollPartCaption(EXEXFILEVIEWLib.ScrollBarEnum.exVScroll,
EXEXFILEVIEWLib.ScrollPartEnum.exRightB1Part, "<img></img>2");
```

```
axExFileView1.EndUpdate();
```

```
private void axExFileView1_ScrollButtonClick(object sender,
AxEXEXFILEVIEWLib._IExFileViewEvents_ScrollButtonClickEvent e)
{
   MessageBox.Show(e.scrollPart.ToString());
}
```

The following C++ sample displays the identifier of the scroll's button being clicked:

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

```
void OnScrollButtonClickExFileView1(long ScrollPart)
{
   CString strFormat;
   strFormat.Format( _T("%i"), ScrollPart );
   MessageBox( strFormat );
}
```

The following VFP sample displays the identifier of the scroll's button being clicked:

```
With thisform.ExFileView1
   .BeginUpdate
      .ScrollPartVisible(0, bitor(32768,32)) = .t.
      .ScrollPartCaption(0,32768) = "<img></img>1"
      .ScrollPartCaption(0, 32) = "<img></img>2"
   .EndUpdate
EndWith
```

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

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

event Search (State as SearchStateEnum)

Occurs when searching files starts or ends

Type	Description
State as SearchStateEnum	A SearchStateEnum expression that indicates the searching state.

The Search event is fired when searching files starts or ends. Use the Search property to search for files. Use the Add method to add rules to customize the found items. Use the StopSearch method to stop immediately searching the files. Use the Get property to get the collection of found files.

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

C#

```
private void Search(object sender,exontrol.EXFILEVIEWLib.SearchStateEnum State)
{
}
```

VB

```
Private Sub Search(ByVal sender As System.Object,ByVal State As
exontrol.EXFILEVIEWLib.SearchStateEnum) Handles Search
End Sub
```

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

C#

```
private void Search(object sender,
AxEXFILEVIEWLib._IExFileViewEvents_SearchEvent e)
{
}
```

C++

```
void OnSearch(long State)
{
}
```

C++ Builder

```
void __fastcall Search(TObject *Sender,Exfileviewlib_tlb::SearchStateEnum State)
{
}
```

Delphi

```
procedure Search(ASender: TObject; State : SearchStateEnum);
begin
```

```
end;
```

Delphi 8 (.NET only)

```
procedure Search(sender: System.Object; e:
AxEXFILEVIEWLib._IExFileViewEvents_SearchEvent);
begin
end;
```

Powe…

```
begin event Search(long State)
end event Search
```

VB.NET

```
Private Sub Search(ByVal sender As System.Object, ByVal e As
AxEXFILEVIEWLib._IExFileViewEvents_SearchEvent) Handles Search
End Sub
```

VB6

```
Private Sub Search(ByVal State As EXFILEVIEWLibCtl.SearchStateEnum)
End Sub
```

VBA

```
Private Sub Search(ByVal State As Long)
End Sub
```

VFP

```
LPARAMETERS State
```

Xbas…

```
PROCEDURE OnSearch(oExFileView,State)
RETURN
```

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

Java…

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

VBSc…

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

Visual Data…

```
Procedure OnComSearch OLESearchStateEnum llState
   Forward Send OnComSearch llState
```

```
End_Procedure
```

Visual Objects

```
METHOD OCX_Search(State) CLASS MainDialog
RETURN NIL
```

X++

```
void onEvent_Search(int _State)
{
}
```

XBasic

```
function Search as v (State as OLE::Exontrol.ExFileView.1::SearchStateEnum)
end function
```

dBASE

```
function nativeObject_Search(State)
return
```

The following VB sample displays a message when searching the files starts or ends:

```
Private Sub ExFileView1_Search(ByVal State As EXFILEVIEWLibCtl.SearchStateEnum)
   Select Case State
      Case EXFILEVIEWLibCtl.SearchStateEnum.StartSearching
         Debug.Print "Start searching '" & ExFileView1.Search & "' files"
      Case EXFILEVIEWLibCtl.SearchStateEnum.EndSearching
         Debug.Print "End searching"
   End Select
End Sub
```

The following C++ sample displays a message when searching the files starts or ends:

```
void OnSearchExfileview1(long State)
{
   switch ( State )
   {
      case 0: /*StartSearching*/
      {
         OutputDebugString( "Start searching ..." );
         break;
      }
      case 1: /*EndSearching*/
```

```
        {
            OutputDebugString( "End searching ..." );
            break;
        }
    }
}
```

The following VB.NET sample displays a message when searching the files starts or ends:

```
Private Sub AxExFileView1_SearchEvent(ByVal sender As Object, ByVal e As
AxEXFILEVIEWLib._IExFileViewEvents_SearchEvent) Handles AxExFileView1.SearchEvent
    Select Case e.state
        Case EXFILEVIEWLib.SearchStateEnum.StartSearching
            Debug.Print("Start searching '" & AxExFileView1.Search & "' files")
        Case EXFILEVIEWLib.SearchStateEnum.EndSearching
            Debug.Print("End searching")
    End Select
End Sub
```

The following C# sample displays a message when searching the files starts or ends:

```
private void axExFileView1_SearchEvent(object sender,
AxEXFILEVIEWLib._IExFileViewEvents_SearchEvent e)
{
    switch (e.state)
    {
        case EXFILEVIEWLib.SearchStateEnum.StartSearching:
        {
            System.Diagnostics.Debug.WriteLine("Start searching '" + axExFileView1.Search + "'
files.");
            break;
        }
        case EXFILEVIEWLib.SearchStateEnum.EndSearching:
        {
            System.Diagnostics.Debug.WriteLine("End searching");
            break;
        }
    }
```

```
}
```

The following VFP sample displays a message when searching the files starts or ends:

```
*** ActiveX Control Event ***
LPARAMETERS state

do case
   case state = 0
      wait window nowait "Start searching"
   case state = 1
      wait window nowait "End searching"
endcase
```

event StateChange (State as StateChangeEnum)

Fired while the control's state has been changed.

Type	Description
State as StateChangeEnum	A StateChangeEnum value that indicates the new control's state.

The StateChange event notifies your application that the current selection is changed. Use Get property to retrieves the collection of selected files or folders. Use the Name property to specify the name of the file or the folder. Use the Folder property to specify whether an item holds a file or a folder. Use the BrowseFolderPath property to specify the path to the browsed folder.

The StateChange event is fired in one of the following situations:

- (0) RenameState A file was renamed, only if the AllowRename property is true.
- (1) SetFocusState, The control gains the focus.
- (2) KillFocusState, The control loses the focus.
- (3) SelChangeState, The current selection is changed.
- (4) BrowseChangeState, The control browses a new folder. Use the BrowseFolderPath property to specify the path to the browsed folder.
- (5) RefreshState, The control did refresh the list of items.
- (6) UpdateChangeState, The control was notified by system, that a file/folder was changed, added, moved, or removed.
- (7) BeforeFilterChangeState, The control starts filtering the items.
- (8) AfterFilterChangeState, The control ends filtering the items

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

C#

```
private void StateChange(object
sender,exontrol.EXFILEVIEWLib.StateChangeEnum State)
{
}
```

VB

```
Private Sub StateChange(ByVal sender As System.Object,ByVal State As
exontrol.EXFILEVIEWLib.StateChangeEnum) Handles StateChange
End Sub
```

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

C#

```
private void StateChange(object sender,
```

```
AxEXFILEVIEWLib._IExFileViewEvents_StateChangeEvent e)
{
}
```

C++

```
void OnStateChange(long State)
{
}
```

C++ Builder

```
void __fastcall StateChange(TObject *Sender,Exfileviewlib_tlb::StateChangeEnum State)
{
}
```

Delphi

```
procedure StateChange(ASender: TObject; State : StateChangeEnum);
begin
end;
```

Delphi 8 (.NET only)

```
procedure StateChange(sender: System.Object; e: AxEXFILEVIEWLib._IExFileViewEvents_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 AxEXFILEVIEWLib._IExFileViewEvents_StateChangeEvent) Handles StateChange
End Sub
```

VB6

```
Private Sub StateChange(ByVal State As EXFILEVIEWLibCtl.StateChangeEnum)
End Sub
```

VBA

```
Private Sub StateChange(ByVal State As Long)
End Sub
```

VFP

```
LPARAMETERS State
```

Xbas…

```
PROCEDURE OnStateChange(oExFileView,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.ExFileView.1::StateChangeEnum)
end function
```

dBASE

```
function nativeObject_StateChange(State)
return
```

Please check the list of all states in the StateChangeEnum type.

The following VB sample enumerates the selected items:

```
Private Sub ExFileView1_StateChange(ByVal State As EXFILEVIEWLibCtl.StateChangeEnum)
   If State = SelChangeState Then
```

```
        Dim fs As Files, f As File
        Set fs = ExFileView1.Get(SelItems)
        For Each f In fs
            Debug.Print f.Name
        Next
    End If
End Sub
```

The following C++ sample enumerates the selected items:

```
void OnStateChangeExfileview1(long State)
{
    switch ( State )
    {
        case 0: /*StartSearching*/
        {
            OutputDebugString( "Start searching" );
            break;
        }
        case 1: /*EndSearching*/
        {
            OutputDebugString( "End searching" );
            break;
        }
    }
}
```

The following VB.NET sample enumerates the selected items:

```
Private Sub AxExFileView1_StateChange(ByVal sender As Object, ByVal e As
AxEXFILEVIEWLib._IExFileViewEvents_StateChangeEvent) Handles
AxExFileView1.StateChange
    Select Case e.state
        Case EXFILEVIEWLib.StateChangeEnum.SelChangeState
            With AxExFileView1.get_Get(EXFILEVIEWLib.TypeEnum.SelItems)
                Dim i As Integer
                For i = 0 To .Count - 1
                    With .Item(i)
```

```
                Debug.WriteLine(.Name)
            End With
        Next
    End With
  End Select
End Sub
```

The following C# sample enumerates the selected items:

```
private void axExFileView1_StateChange(object sender,
AxEXFILEVIEWLib._IExFileViewEvents_StateChangeEvent e)
{
  switch (e.state)
  {
    case EXFILEVIEWLib.StateChangeEnum.SelChangeState:
    {
      EXFILEVIEWLib.Files files =
axExFileView1.get_Get(EXFILEVIEWLib.TypeEnum.SelItems);
      for (int i = 0; i < files.Count; i++)
      {
        EXFILEVIEWLib.File file = files[i];
        System.Diagnostics.Debug.WriteLine(file.Name);
      }
      break;
    }
  }
}
```

The following VFP sample enumerates the selected items:

```
*** ActiveX Control Event ***
LPARAMETERS state

do case

 case state = 3 && SelChangeState
 with thisform.ExFileView1.Get( 0 ) && SelItems
   local i
```

```
    for i = 0 to .Count - 1
        with .Item(i)
            wait window nowait .Name
        endwith
    next
 endwith

endcase
```

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 .