



ExListBar

The Exontrol's ExListBar component, an accurate reproduction of the Microsoft Outlook Bar, provides an intuitive user-interface when large amounts of information need to be presented. The ExListBar 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.

Features include:

- **WYSWYG Template/Layout Editor** support.
- **Skinnable Interface** support (ability to apply a skin to any background part)
- **Shortcut bar** support (Ability to group the groups of items, in the shortcut bar).
- Multi-Lines HTML **ToolTip** support
- Any item supports **built-in HTML tags**.
- New text decorations support for HTML captions, like outlined characters, shadow,...
- Custom size pictures support.
- Gradient Colors Support.
- Vertical or horizontal orientation
- Nice animation when group is selected
- Unlimited color and font attributes options for items or groups (tabs)
- Ability to scroll the items into a group
- Auto Arrange Support
- Ability to display a picture on the groups background
- Multiple type of alignments for items, groups
- Small or large icons support
- Mouse wheel support
- Keyboard support



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

How to get support?

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

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

- Check out the samples - they are here to provide some quick info on how things should be done
- Check out the how-to questions using the [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 types. Use the [Alignment](#) property to align the item. Use the [Alignment](#) property to align the caption of the group.

Name	Value	Description
exLeft	0	The object is left aligned.
exCenter	1	The object is center aligned.
exRight	2	The object is right aligned.
exMiddle	4	The object's icon and object's caption are center aligned. The object's icon is displayed on top, and the object's caption is bottom displayed.

constants AppearanceEnum

Specifies the control's border, or group's appearance. Use the [Appearance](#) property to specify the control's appearance.

Name	Value	Description
exNone	0	No border
exSingle	1	Single
exRaised	2	Raised
exPop	3	Pop
exDrop	4	Drop
exShadow	5	Shadow
exInset	6	Inset

constants BackgroundPartEnum

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

Name	Value	Description
exScrollUpUp	0	Specifies the visual appearance for the scroll up button, when it is up.
exScrollUpDown	1	Specifies the visual appearance for the scroll up button, when it is down.
exScrollDownUp	2	Specifies the visual appearance for the scroll down button, when it is up.
exScrollDownDown	3	Specifies the visual appearance for the scroll down button, when it is down.
exSelectItem	4	Specifies the visual appearance for the selected item. Use the SelectItemType property to specify whether the control marks the selected item.
exHightlightItem	5	Specifies the visual appearance when the cursor hovers an item. The HighlightItemType property indicates whether the item from the cursor is highlighted.
exToolTipAppearance	64	Indicates the visual appearance of the borders of the tooltips. Use the ToolTipPopDelay property specifies the period in ms of time the ToolTip remains visible if the mouse pointer is stationary within a control. Use the ToolTipWidth property to specify the width of the tooltip window. The ToolTipDelay property specifies the time in ms that passes before the ToolTip appears. Use the ShowToolTip method to display a custom tooltip.
exToolTipBackColor	65	Specifies the tooltip's background color.
exToolTipForeColor	66	Specifies the tooltip's foreground color.

constants CaptionFormatEnum

Defines how the cell's caption is painted. Use the [CaptionFormat](#) property to specify whether the item's caption supports built-in HTML format. Use the [CaptionFormat](#) property to specify whether the group's caption supports built-in HTML format.

Name	Value	Description
exText	0	exText
		<p>The control uses built-in HTML tags to display the caption using HTML format. The control supports the following HTML tags:</p> <ul style="list-style-type: none">• bold • <u> underline </u>• <s> strikeout </s>• <i> italic </i>• <fgcolor = FF0000> fgcolor </fgcolor>• <bgcolor = FF0000> bgcolor </bgcolor>•
 breaks a line.• <solidline> draws a solid line• <dotline> draws a dotted line• <upline> draws the line to the top of the text line• <r> aligns the rest of the text line to the right side• number inserts an icon inside the item's caption. The number indicates the index of the icon being inserted.• key[:width] inserts a custom size picture being 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.• text displays portions of text with a different font and/or different size. For instance, the bit draws the bit text using the Tahoma font, on size 12 pt. If the name of the font is missing, and instead size is present, the current font is used with a different size.

For instance, `bit` displays the bit text using the current font, but with a different size.

- **`<a id;options>`** [anchor](#) ``, specifies hyperlinks in your text. An anchor is a piece of text or some other object (for example an image) which marks the beginning and/or the end of a hypertext link. The `<A>` element is used to mark that piece of text (or inline image), and to give its hypertextual relationship to other documents. The control fires the [AnchorClick](#) event when the user clicks an anchor element. Use the [FormatAnchor](#) property to specify the visual effect for anchor elements. For instance, if the user clicks the anchor `<a1>anchor`, the control fires the AnchorClick event, where the AnchorID parameter is 1, and the Options parameter is empty. Also, if the user clicks the anchor `<a 1;youreextradata>anchor`, the AnchorID parameter of the AnchorClick event is 1, and the Options parameter is "youreextradata". Use the [AnchorFromPoint](#) property to retrieve the identifier of the anchor element from the cursor. Use the [ShowToolTip](#) method to display a custom tooltip.

exHTML

1

Also, newer HTML format supports decorative text like follows:

- **`<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

constants HighLightItemEnum

Specifies the way how the control marks the highlighted items. Use the [HighlightItemType](#) property to specify the way how the control highlights the item from the cursor.

Name	Value	Description
exNoHighlight	0	The control draws no highlighted item
exCaption	1	The item's caption is highlighted.
exIcon	2	The item's icon is highlighted.
exUnion	3	The item's drawing area is highlighted.
exFull	16	The exFull flag can be combined with any exCaption, exIcon or exUnion flag to specify that the item is selectable whenever the cursor hovers any part of the item, not only its icon or caption part

constants OrientationEnum

Specifies the control's orientation. Use the [Orientation](#) property to specify the control's orientation.

Name	Value	Description
exVertical	0	The control is vertically oriented.
exHorizontal	1	The control is horizontally oriented.

constants PictureBoxEnum

Specifies how the picture is displayed on the control's background. Use the PictureBox 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 SelectItemEnum

Specifies the way how the control displays the selected item. Use the [SelectItemType](#) property to specify the way how the control displays the selected item.

Name	Value	Description
exNoSelect	0	The control draws no selected items
exSelectPush	1	The control draws a push button around the selected item
exSelectPop	2	The control draws a pop button around the selected item

constants UIVisualThemeEnum

The UIVisualThemeEnum expression specifies the UI parts that the control can shown using the current visual theme. The [UseVisualTheme](#) property specifies whether the UI parts of the control are displayed using the current visual theme.

Name	Value	Description
exNoVisualTheme	0	exNoVisualTheme
exDefaultVisualTheme	16777215	exDefaultVisualTheme
explorerBarVisualTheme	512	explorerBarVisualTheme

Appearance object

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

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

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

Adds or replaces a skin object to the control.

Type	Description
ID as Long	<p>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.</p>
Skin as Variant	<p>A string expression that indicates:</p> <ol style="list-style-type: none">1. an Windows XP Theme part, it should start with "XP:". For instance the "XP:Header 1 2" indicates the part 1 of the Header class in the state 2, in the current Windows XP theme. In this case the format of the Skin parameter should be: "XP: Control/ClassName Part State" where the ClassName defines the window/control class name in the Windows XP Theme, the Part indicates a long expression that defines the part, and the State indicates the state like listed at the end of the document. This option is available only on Windows XP that supports Themes API.2. a copy of another skin with different coordinates, if it begins with "CP:" . For instance, you may need to display a specified skin on a smaller rectangle. In this case, the string starts with "CP:", and in may looks like "CP:n l t r b", where the n is the identifier being copied, the l, t, r, and b indicates the left, top, right and bottom coordinates being used to adjust the rectangle where the copied skin is displayed. For instance, the "CP:1 4 0 -4 0", indicates that the skin is displayed on a smaller rectangle like follows: Let's say that the control request to paint the {10, 10, 30, 20} area, a rectangle with width of 20 pixels, and height of 10 pixels, the skin will be displayed on the {14,10,26,20} as each coordinates in the "CP" syntax is added to the displayed rectangle. This way you can apply different effects to your objects in your control.



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

A byte[] or safe arrays of VT_I1 or VT_UI1 expression that indicates the content of the EBN file. You can use this option when using the EBN file directly in the resources of the project. For instance, the VB6 provides the LoadResData to get the safe array o bytes for specified resource, while in VB/NET or C# the internal class Resources provides definitions for all files being inserted. (ResourceManager.GetObject("ebn", resourceCulture)).

Return

Description

Boolean

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


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



The identifier you choose for the skin is very important to be used in the background properties like explained below. 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:

- headers of the groups, [BackColorGroup](#) property, [SelBackColorGroup](#) property, [BackColor](#) property
- item, [BackColor](#) property
- scroll bars, selected item, [Background](#) property

For instance, the following VB sample changes the visual appearance for group headers. The [BackColorGroup](#) property indicates the indicates the default group's 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 BackColorGroup property to indicates the index of the skin that we want to use. The sample applies the skin "".

```
With ListBar1
```

```
    .VisualAppearance.Add 1, "D:\Temp\ExListBar.Help\tabdown1.ebn"
```

```
    .BackColorGroup = &H1000000
```

```
End With
```

The following C++ sample changes the visual appearance for group headers:

```
#include "Appearance.h"
m_listbar.GetVisualAppearance().Add( 1, COleVariant(
"D:\\Temp\\ExListBar.Help\\tabup1.ebn" ) );
m_listbar.SetBackColorGroup( 0x1000000 );
```

The following VB.NET sample changes the visual appearance for group headers:

```
With AxListBar1
    .VisualAppearance.Add(1, "D:\\Temp\\ExListBar.Help\\tabup1.ebn")
    .Template = "BackColorGroup = 16777216"
End With
```

The following C# sample changes the visual appearance for group headers:

```
axListBar1.VisualAppearance.Add(1, "D:\\Temp\\ExListBar.Help\\tabup1.ebn");
axListBar1.Template = "BackColorGroup = 16777216";
```

The following VFP sample changes the visual appearance for group headers:

```
With thisform.ListBar1
    .VisualAppearance.Add(1, "D:\\Temp\\ExListBar.Help\\tabup1.ebn")
    .BackColorGroup = 16777216
EndWith
```

where the 16777216 value represents 0x1000000 in hexadecimal.

The [screen show](#) was generated using the following template:

```
BeginUpdate
' Properties, Objects, Methods          ' Comments

Images("gBJJgBggAAwAAgACEKAD/hz/EMNh8TIRNGwAjEZAEXjAojJAjlgjIBAEijUIk8pIUrlktl
Images("gBJJgBggAAwAAgACEKAD/hz/EMNh8TIRNGwAjEZAEXjAojJAjlgjIBAEijUIk8pIUrlktl
Images("gBJJgBggAAkGAAQhIAf8Nf4hhkOiRCJo2AEXjAAi0XFEYIEYhUXAIAEEZi8hk0pIUrlktl
```

VisualAppearance

{

' Headers

Add(1,"gBFLBCJwBAEHhEJAEGg4BMwHg6AADACAxRDAMgBQKAAzQFAYZhxBaERiGIZ4JhL

Add(2,"gBFLBCJwBAEHhEJAEGg4BWYGg6AADACAxRDAMgBQKAAzQFAYZhxBaERiGIZ4JhU

' Scroll Up, Down

Add(3,"gBFLBCJwBAEHhEJAEGg4BWMIQAAYAQGKIYBkAKBQAGaAoDDMOILQiMQxDPBmK

Add(4,"gBFLBCJwBAEHhEJAEGg4BW8IQAAYAQGKIYBkAKBQAGaAoDDMOILQiMQxDPBmK,

' Select, Highligth

Add(5,

"gBFLBCJwBAEHhEJAEGg4Ba4Fg6AABACAxWgKBADQKAAYDIKsEQGGIZRhhGlwAgaFIXQKI
)

Add(6,"gBFLBCJwBAEHhEJAEGg4BagFg6AADACAxRDAMgBQKAAzQFAYZhxBaERiGIZ4JhUA

}

Background(0) = 50331648

Background(1) = 50331648

Background(2) = 67108864

Background(3) = 67108864

Background(4) = 83886080

Background(5) = 100663296

BackColorgroup = 16777216

SelBackColorGroup = 33554432

SelForeColorGroup = 0

DelayScroll = 200

' Specifies the scroll delay. When the user selects a new

group.
SelectItemType = 1
GroupHeight = 40
MarkSelectGroup = True

Groups ' Gets the groups collection

{ ' Opens the groups context

"

Outlook **Shortcuts** ' Appends a new group to Groups collection

{

Alignment = 0

CaptionFormat = 1

Image =

"gBHJJGHA5MIgAEIe4AAAFhwQiAbCAbigbEsWGAIGA7Eo7HcbIowlpFHZQkZQKA7IsplErlBl

ItemHeight = 36 ' Specifies the item's height

AddItem("Outlook Today") ' Appends a new item to group object

{ ' Opens the item context

Image =

"gBHJJGHA5MIqAAXAD3AENhohzhpmhqZhrMhr/h0QGcQM0QTMQZkQf8QAESGcSM0STM

' Assign an icon to the group

' Assigns an icon to

the item

Alignment = 4 ' Specifies the item's alignment such as exMiddle

} ' Closes the item context

AddItem("Inbox") ' Appends a new item to group object

{ ' Opens the item context

ToolTip = "Click the Inbox folder to go to your e-mail, or click a news server name
or specific newsgroup to visit newsgroups."

Image = 3 ' Assigns an icon to the item

Alignment = 4 ' Specifies the item's alignment such as exMiddle

Bold = True ' Bolds the item

} ' Closes the item context

AddItem("Calendar") ' Appends a new item to group object

{ ' Opens the item context

Image = 1 ' Assigns an icon to the item

Alignment = 4 ' Specifies the item's alignment such as exMiddle

} ' Closes the item context

```

AddItem("Contacts")          ' Appends a new item to group object
{
    Image = 2                ' Assigns an icon to the item
    Alignment = 4             ' Specifies the item's alignment such as exMiddle
}
AddItem("Tasks")             ' Appends a new item to group object
{
    Image = 3                ' Assigns an icon to the item
    Alignment = 4             ' Specifies the item's alignment such as exMiddle
}
AddItem("Notes")             ' Appends a new item to group object
{
    Image = 1                ' Assigns an icon to the item
    Alignment = 4             ' Specifies the item's alignment such as exMiddle
}
AddItem("Deleted Items")     ' Appends a new item to group object
{
    Image = 2                ' Assigns an icon to the item
    Alignment = 4             ' Specifies the item's alignment such as exMiddle
}
}
"

```

```

My Shortcuts"               ' Appends a new group to Groups collection
{
    CaptionFormat = 1
    Alignment = 0
    Image =
"gBHJJGHA5MliAEIe4AAAFhwFIJpApWPoFNSbCAPB4QJhLCBWKoQNRpCB+V8IXIQRDDDC

```

```

AddItem("Draft", 6)         ' Adds a new item
{
    Alignment = 0            ' Aligns the item to left
}
AddItem("Outbox", 7)        ' Adds a new item
{
    Alignment = 0            ' Aligns the item to left
}

```

```

AddItem("Sent Items", 8)      ' Adds a new item
{                             ' Opens the item context
    Alignment = 0             ' Aligns the item to left
}                             ' Closes the item context
AddItem("Journal", 9)        ' Adds a new item
{                             ' Opens the item context
    Alignment = 0             ' Aligns the item to left
}                             ' Closes the item context
AddItem("Outlook Update", 10) ' Adds a new item
{                             ' Opens the item context
    Alignment = 0             ' Aligns the item to left
}                             ' Closes the item context
}                             ' closes the group context
"
Other Shortcuts"             ' Appends a new group to Groups collection
{                             ' Opens the group context
    CaptionFormat = 1
    Alignment = 0
    Image =
"gBHJJGHA5MIgAEIe4AAAFhwQiAbCAFDcVEolCEXEowjg7GAbHY7CEhHZFkRFIBQIoQKEtLZ

    AddItem("My Computer", 5)      ' Adds a new item and associates an icon to it
    AddItem("My documents", 6)     ' Adds a new item and associates an icon to it
    AddItem("Favorites", 7)        ' Adds a new item and associates an icon to it
}                             ' closes the group context
}
EndUpdate

```

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

Control/ClassName	Part	States
		CBS_UNCHECKED
		1 CBS_UNCHECKE
		CBS_UNCHECKED
		= 3
		CBS_UNCHECKED

BUTTON	BP_CHECKBOX = 3	= 4 CBS_CHECKEDH 5 CBS_CHECKEDPR CBS_CHECKEDPR CBS_CHECKEDDIS CBS_MIXEDNORM CBS_MIXEDHOT = CBS_MIXEDPRES CBS_MIXEDDISAB
	BP_GROUPBOX = 4	GBS_NORMAL = 1 GBS_DISABLED =
	BP_PUSHBUTTON = 1	PBS_NORMAL = 1 = 2 PBS_PRESSED PBS_DISABLED = PBS_DEFAULTED :
	BP_RADIOBUTTON = 2	RBS_UNCHECKED 1 RBS_UNCHECKE RBS_UNCHECKED = 3 RBS_UNCHECKED = 4 RBS_CHECKED 5 RBS_CHECKEDH RBS_CHECKEDPR RBS_CHECKEDDIS
	BP_USERBUTTON = 5	
CLOCK	CLP_TIME = 1	CLS_NORMAL = 1 CBXS_NORMAL = CBXS_HOT = 2 CBXS_PRESSED = CBXS_DISABLED :
COMBOBOX	CP_DROPDOWNBUTTON = 1	
EDIT	EP_CARET = 2	
	EP_EDITTEXT = 1	ETS_NORMAL = 1 2 ETS_SELECTED ETS_DISABLED = ETS_FOCUSED = ETS_READONLY = ETS_ASSIST = 7
EXPLORERBAR	EBP_HEADERBACKGROUND = 1	
	EBP_HEADERCLOSE = 2	EBHC_NORMAL = EBHC_HOT = 2 EBHC_PRESSED = EBHP_NORMAL =

EBP_HEADERPIN = 3

EBP_IEBARMENU = 4

EBP_NORMALGROUPBACKGROUND = 5

EBP_NORMALGROUPCOLLAPSE = 6

EBP_NORMALGROUPEXPAND = 7

EBP_NORMALGROUPHEAD = 8

EBP_SPECIALGROUPBACKGROUND = 9

EBP_SPECIALGROUPCOLLAPSE = 10

EBP_SPECIALGROUPEXPAND = 11

EBP_SPECIALGROUPHEAD = 12

HEADER

HP_HEADERITEM = 1

HP_HEADERITEMLEFT = 2

HP_HEADERITEMRIGHT = 3

HP_HEADERSORTARROW = 4

LISTVIEW

LVP_EMPTYTEXT = 5

LVP_LISTDETAIL = 3

LVP_LISTGROUP = 2

LVP_LISTITEM = 1

EBHP_HOT = 2
EBHP_PRESSED =
EBHP_SELECTED
4 EBHP_SELECTED
EBHP_SELECTED
6

EBM_NORMAL = 1
= 2 EBM_PRESSED

EBNGC_NORMAL
EBNGC_HOT = 2
EBNGC_PRESSED
EBNGE_NORMAL :
EBNGE_HOT = 2
EBNGE_PRESSED

EBSGC_NORMAL :
EBSGC_HOT = 2
EBSGC_PRESSED
EBSGE_NORMAL :
EBSGE_HOT = 2
EBSGE_PRESSED

HIS_NORMAL = 1
2 HIS_PRESSED =
HILS_NORMAL = 1
= 2 HILS_PRESSED
HIRS_NORMAL = 1
= 2 HIRS_PRESSED
HSAS_SORTEDUP
HSAS_SORTEDDC

LIS_NORMAL = 1
2 LIS_SELECTED :
LIS_DISABLED = 4
LIS_SELECTEDNO
5

LVP_LISTSORTEDDETAIL = 4

MENU

MP_MENUBARDROPDOWN = 4

MP_MENUBARITEM = 3

MP_CHEVRON = 5

MP_MENUDROPDOWN = 2

MP_MENUITEM = 1

MP_SEPARATOR = 6

MENUBAND

MDP_NEWAPPBUTTON = 1

MDP_SEPERATOR = 2

PAGE

PGRP_DOWN = 2

PGRP_DOWNHORZ = 4

PGRP_UP = 1

PGRP_UPHORZ = 3

PROGRESS

PP_BAR = 1

PP_BARVERT = 2

MS_NORMAL = 1

MS_SELECTED = 2

MS_DEMOTED = 3

MS_NORMAL = 1

MS_SELECTED = 2

MS_DEMOTED = 3

MS_NORMAL = 1

MS_SELECTED = 2

MS_DEMOTED = 3

MS_NORMAL = 1

MS_SELECTED = 2

MS_DEMOTED = 3

MS_NORMAL = 1

MS_SELECTED = 2

MS_DEMOTED = 3

MS_NORMAL = 1

MS_SELECTED = 2

MS_DEMOTED = 3

MDS_NORMAL = 1

= 2 MDS_PRESSED

MDS_DISABLED =

MDS_CHECKED =

MDS_HOTCHECKE

DNS_NORMAL = 1

= 2 DNS_PRESSED

DNS_DISABLED =

DNHZS_NORMAL =

DNHZS_HOT = 2

DNHZS_PRESSED

DNHZS_DISABLED

UPS_NORMAL = 1

= 2 UPS_PRESSED

UPS_DISABLED =

UPHZS_NORMAL =

UPHZS_HOT = 2

UPHZS_PRESSED

UPHZS_DISABLED

PP_CHUNK = 3

PP_CHUNKVERT = 4

REBAR

RP_BAND = 3

RP_CHEVRON = 4

RP_CHEVRONVERT = 5

RP_GRIPPER = 1

RP_GRIPPERVERT = 2

CHEVS_NORMAL =

CHEVS_HOT = 2

CHEVS_PRESSED

ABS_DOWNDISAB

ABS_DOWNHOT,

ABS_DOWNNORM

ABS_DOWNPRESS

ABS_UPDISABLED

ABS_UPHOT,

ABS_UPNORMAL,

ABS_UPPRESSED,

ABS_LEFTDISABLI

ABS_LEFTHOT,

ABS_LEFTNORMA

ABS_LEFTPRESSE

ABS_RIGHTDISAB

ABS_RIGHTHOT,

ABS_RIGHTNORM

ABS_RIGHTPRESS

SCROLLBAR

SBP_ARROWBTN = 1

SBP_GRIPPERHORZ = 8

SBP_GRIPPERVERT = 9

SBP_LOWERTRACKHORZ = 4

SBP_LOWERTRACKVERT = 6

SBP_THUMBBTNHORZ = 2

SCRBS_NORMAL :

SCRBS_HOT = 2

SCRBS_PRESSED

SCRBS_DISABLED

SCRBS_NORMAL :

SCRBS_HOT = 2

SCRBS_PRESSED

SCRBS_DISABLED

SCRBS_NORMAL :

SCRBS_HOT = 2

SCRBS_PRESSED

SCRBS_DISABLED

SCRBS_NORMAL :

SBP_THUMBBTNVERT = 3

SBP_UPPERTRACKHORZ = 5

SBP_UPPERTRACKVERT = 7

SBP_SIZEBOX = 10

SPIN

SPNP_DOWN = 2

SPNP_DOWNHORZ = 4

SPNP_UP = 1

SPNP_UPHORZ = 3

STARTPANEL

SPP_LOGOFF = 8

SPP_LOGOFFBUTTONS = 9

SPP_MOREPROGRAMS = 2

SPP_MOREPROGRAMSARROW = 3

SPP_PLACESLIST = 6

SPP_PLACESLISTSEPARATOR = 7

SPP_PREVIEW = 11

SPP_PROGLIST = 4

SPP_PROGLISTSEPARATOR = 5

SPP_USERPANE = 1

SCRBS_HOT = 2

SCRBS_PRESSED

SCRBS_DISABLED

SCRBS_NORMAL :

SCRBS_HOT = 2

SCRBS_PRESSED

SCRBS_DISABLED

SCRBS_NORMAL :

SCRBS_HOT = 2

SCRBS_PRESSED

SCRBS_DISABLED

SZB_RIGHTALIGN

SZB_LEFTALIGN =

DNS_NORMAL = 1

= 2 DNS_PRESSED

DNS_DISABLED =

DNHZZ_NORMAL :

DNHZZ_HOT = 2

DNHZZ_PRESSED

DNHZZ_DISABLED

UPS_NORMAL = 1

= 2 UPS_PRESSED

UPS_DISABLED =

UPHZZ_NORMAL :

UPHZZ_HOT = 2

UPHZZ_PRESSED

UPHZZ_DISABLED

SPLS_NORMAL =

SPLS_HOT = 2

SPLS_PRESSED =

SPS_NORMAL = 1

= 2 SPS_PRESSED

STATUS

SPP_USERPICTURE = 10

SP_GRIPPER = 3

SP_PANE = 1

TAB

SP_GRIPPERPANE = 2

TABP_BODY = 10

TABP_PANE = 9

TABP_TABITEM = 1

TABP_TABITEMBOTHEDGE = 4

TABP_TABITEMLEFTEDGE = 2

TABP_TABITEMRIGHTEDGE = 3

TABP_TOPTABITEM = 5

TABP_TOPTABITEMBOTHEDGE = 8

TABP_TOPTABITEMLEFTEDGE = 6

TIS_NORMAL = 1

2 TIS_SELECTED :

TIS_DISABLED = 4

TIS_FOCUSED = 5

TIBES_NORMAL =

TIBES_HOT = 2

TIBES_SELECTED

TIBES_DISABLED

TIBES_FOCUSED :

TILES_NORMAL =

TILES_HOT = 2

TILES_SELECTED

TILES_DISABLED :

TILES_FOCUSED :

TIRES_NORMAL =

TIRES_HOT = 2

TIRES_SELECTED

TIRES_DISABLED

TIRES_FOCUSED :

TTIS_NORMAL = 1

= 2 TTIS_SELECTE

TTIS_DISABLED =

TTIS_FOCUSED =

TTIBES_NORMAL :

TTIBES_HOT = 2

TTIBES_SELECTEI

TTIBES_DISABLED

TTIBES_FOCUSED

TTILES_NORMAL :

TTILES_HOT = 2

TTILES_SELECTEI

TTILES_DISABLED

TTILES_FOCUSED

TTIRES_NORMAL

TABP_TOPTABITEMRIGHTEDGE = 7

TTIRES_HOT = 2
TTIRES_SELECTE
TTIRES_DISABLED
TTIRES_FOCUSEL

TASKBAND

TDP_GROUPCOUNT = 1

TDP_FLASHBUTTON = 2

TDP_FLASHBUTTONGROUPMENU = 3

TASKBAR

TBP_BACKGROUNDBOTTOM = 1

TBP_BACKGROUNDLEFT = 4

TBP_BACKGROUNDRIGHT = 2

TBP_BACKGROUNDTOP = 3

TBP_SIZINGBARBOTTOM = 5

TBP_SIZINGBARBOTTOMLEFT = 8

TBP_SIZINGBARRIGHT = 6

TBP_SIZINGBARTOP = 7

TOOLBAR

TP_BUTTON = 1

TP_DROPDOWNBUTTON = 2

TP_SPLITBUTTON = 3

TP_SPLITBUTTONDROPDOWN = 4

TP_SEPARATOR = 5

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

TP_SEPARATORVERT = 6

TOOLTIP

TTP_BALLOON = 3

TTP_BALLOONTITLE = 4

TTP_CLOSE = 5

TTP_STANDARD = 1

TTP_STANDARDTITLE = 2

TRACKBAR

TKP_THUMB = 3

TKP_THUMBBOTTOM = 4

TKP_THUMBLEFT = 7

TKP_THUMBRIGHT = 8

TKP_THUMBTOP = 5

TS_NORMAL = 1
TS_PRESSED = 3
TS_DISABLED = 4
TS_CHECKED = 5
TS_HOTCHECKED = 6
TTBS_NORMAL = 1
TTBS_LINK = 2
TTBS_NORMAL = 1
TTBS_LINK = 2
TTCS_NORMAL = 1
TTCS_HOT = 2
TTCS_PRESSED = 3
TTSS_NORMAL = 1
TTSS_LINK = 2
TTSS_NORMAL = 1
TTSS_LINK = 2
TUS_NORMAL = 1
TUS_PRESSED = 2
TUS_FOCUSED = 3
TUS_DISABLED = 4
TUBS_NORMAL = 1
TUBS_HOT = 2
TUBS_PRESSED = 3
TUBS_FOCUSED = 4
TUBS_DISABLED = 5
TUVLS_NORMAL = 1
TUVLS_HOT = 2
TUVLS_PRESSED = 3
TUVLS_FOCUSED = 4
TUVLS_DISABLED = 5
TUVRS_NORMAL = 1
TUVRS_HOT = 2
TUVRS_PRESSED = 3
TUVRS_FOCUSED = 4
TUVRS_DISABLED = 5
TUTS_NORMAL = 1
TUTS_HOT = 2
TUTS_PRESSED = 3
TUTS_FOCUSED = 4
TUTS_DISABLED = 5
TUVS_NORMAL = 1

TKP_THUMBVERT = 6

TKP_TICS = 9

TKP_TICSVERT = 10

TKP_TRACK = 1

TKP_TRACKVERT = 2

TRAYNOTIFY

TNP_ANIMBACKGROUND = 2

TNP_BACKGROUND = 1

TREEVIEW

TVP_BRANCH = 3

TVP_GLYPH = 2

TVP_TREEITEM = 1

WINDOW

WP_CAPTION = 1

WP_CAPTIONSIZINGTEMPLATE = 30

WP_CLOSEBUTTON = 18

WP_DIALOG = 29

WP_FRAMEBOTTOM = 9

WP_FRAMEBOTTOMSIZINGTEMPLATE = 36

WP_FRAMELEFT = 7

WP_FRAMELEFTSIZINGTEMPLATE = 32

WP_FRAMERIGHT = 8

WP_FRAMERIGHTSIZINGTEMPLATE = 34

WP_HELPBUTTON = 23

TUVS_HOT = 2

TUVS_PRESSED =

TUVS_FOCUSED =

TUVS_DISABLED =

TSS_NORMAL = 1

TSVS_NORMAL =

TRS_NORMAL = 1

TRVS_NORMAL =

GLPS_CLOSED =

GLPS_OPENED =

TREIS_NORMAL =

TREIS_HOT = 2

TREIS_SELECTED

TREIS_DISABLED

TREIS_SELECTED

= 5

CS_ACTIVE = 1 CS

= 2 CS_DISABLED

CBS_NORMAL = 1

= 2 CBS_PUSHED

CBS_DISABLED =

FS_ACTIVE = 1 FS

= 2

FS_ACTIVE = 1 FS

= 2

FS_ACTIVE = 1 FS

= 2

HBS_NORMAL = 1

= 2 HBS_PUSHED

HBS_DISABLED =

HSS_NORMAL = 1

WP_HORIZSCROLL = 25

WP_HORIZTHUMB = 26

WP_MAX_BUTTON

WP_MAXCAPTION = 5

WP_MDICLOSEBUTTON = 20

WP_MDIHELPBUTTON = 24

WP_MDIMINBUTTON = 16

WP_MDIRESTOREBUTTON = 22

WP_MDISYSBUTTON = 14

WP_MINBUTTON = 15

WP_MINCAPTION = 3

WP_RESTOREBUTTON = 21

WP_SMALLCAPTION = 2

= 2 HSS_PUSHED
HSS_DISABLED =
HTS_NORMAL = 1
2 HTS_PUSHED =
HTS_DISABLED =
MAXBS_NORMAL
MAXBS_HOT = 2
MAXBS_PUSHED =
MAXBS_DISABLED

MXCS_ACTIVE = 1
MXCS_INACTIVE =
MXCS_DISABLED
CBS_NORMAL = 1
= 2 CBS_PUSHED
CBS_DISABLED =
HBS_NORMAL = 1
= 2 HBS_PUSHED
HBS_DISABLED =
MINBS_NORMAL =
MINBS_HOT = 2
MINBS_PUSHED =
MINBS_DISABLED
RBS_NORMAL = 1
= 2 RBS_PUSHED
RBS_DISABLED =
SBS_NORMAL = 1
= 2 SBS_PUSHED
SBS_DISABLED =
MINBS_NORMAL =
MINBS_HOT = 2
MINBS_PUSHED =
MINBS_DISABLED
MNCS_ACTIVE = 1
MNCS_INACTIVE =
MNCS_DISABLED
RBS_NORMAL = 1
= 2 RBS_PUSHED
RBS_DISABLED =
CS_ACTIVE = 1 CS
= 2 CS_DISABLED

WP_SMALLCAPTIONSTIZINGTEMPLATE = 31

WP_SMALLCLOSEBUTTON = 19

WP_SMALLFRAMEBOTTOM = 12

WP_SMALLFRAMEBOTTOMSTIZINGTEMPLATE
= 37

WP_SMALLFRAMELEFT = 10

WP_SMALLFRAMELEFTSTIZINGTEMPLATE =
33

WP_SMALLFRAMERIGHT = 11

WP_SMALLFRAMERIGHTSTIZINGTEMPLATE =
35

WP_SMALLHELPBUTTON

WP_SMALLMAXBUTTON

WP_SMALLMAXCAPTION = 6

WP_SMALLMINCAPTION = 4

WP_SMALLRESTOREBUTTON

WP_SMALLSYSBUTTON

WP_SYSBUTTON = 13

WP_VERTSCROLL = 27

CBS_NORMAL = 1
= 2 CBS_PUSHED
CBS_DISABLED =
FS_ACTIVE = 1 FS
= 2

FS_ACTIVE = 1 FS
= 2

FS_ACTIVE = 1 FS
= 2

HBS_NORMAL = 1
= 2 HBS_PUSHED
HBS_DISABLED =
MAXBS_NORMAL
MAXBS_HOT = 2
MAXBS_PUSHED =
MAXBS_DISABLED

MXCS_ACTIVE = 1
MXCS_INACTIVE =
MXCS_DISABLED

MNCS_ACTIVE = 1
MNCS_INACTIVE =
MNCS_DISABLED

RBS_NORMAL = 1
= 2 RBS_PUSHED
RBS_DISABLED =

SBS_NORMAL = 1
= 2 SBS_PUSHED
SBS_DISABLED =

SBS_NORMAL = 1
= 2 SBS_PUSHED
SBS_DISABLED =

VSS_NORMAL = 1
= 2 VSS_PUSHED

WP_VERTTHUMB = 28

VSS_DISABLED =
VTS_NORMAL = 1
2 VTS_PUSHED =
VTS_DISABLED =

method Appearance.Clear ()

Removes all skins in the control.

Type	Description
------	-------------

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

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

- headers of the groups, [BackColorGroup](#) property, [SelBackColorGroup](#) property, [BackColor](#) property
- item, [ItemBackColor](#) property
- scroll bars, selected item, [Background](#) property

method Appearance.Remove (ID as Long)

Removes a specific skin from the control.

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

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

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

- headers of the groups, [BackColorGroup](#) property, [SelBackColorGroup](#) property, [BackColor](#) property
- item, [ItemBackColor](#) property
- scroll bars, selected item, [Background](#) property

Group object

The Group object holds a collection of Item objects. The [Item](#) property accesses a group object. The Group object supports the following properties and methods:

Name	Description
AddItem	Adds a new item to the group object.
Alignment	Retrieves or sets a value that indicates the caption's alignment.
AutoScroll	Specifies whether the scroll buttons are automatically added.
BackColor	Retrieves or sets the group's background color.
BackColor2	Specifies the color at the ending boundary line of the gradient group's caption.
BackColorList	Retrieves or sets a value that indicates the background color of the list when the group is active.
Bold	Specifies whether the group's caption should appear in bold.
Caption	Specifies the group's caption.
CaptionFormat	Specifies how the group's caption is displayed.
Clear	Clears the items collection.
Count	Gets the count of the items.
EnsureVisibleItem	Ensures that the item fits the group's client area.
ForeColor	Specifies the group's foreground color.
ForeColorList	Retrieves or sets a value that indicates the foreground color of the group's list when it is active.
Image	Specifies the index of the group's icon.
IndentHeaderBottom	Specifies the number of pixels to indent the group's header from the bottom part.
IndentHeaderLeft	Specifies the number of pixels to indent the group's header from the left part.
IndentHeaderRight	Specifies the number of pixels to indent the group's header from the right part.
IndentHeaderTop	Specifies the number of pixels to indent the group's header from the top part.
	Retrieves the index of the object into the Groups

Index	collection..
Italic	Specifies whether the group's caption should appear in italic.
Item	Returns a specific Item from the collection.
ItemByPos	Gets the item given its position.
ItemHeight	Retrieves or sets the item's height.
ItemWidth	Retrieves or sets the item's width. -1 if it's the group's client width.
Picture	Retrieves or sets a graphic to be displayed in the group's list.
PictureDisplay	Retrieves or sets a value that indicates the way how the graphic is displayed on the list's background
Position	Specifies the group's position.
RemoveItem	Removes an item given its index or given its caption.
SelectItem	Specifies the index of item that's selected.
Shortcut	Specifies the name of the shortcut which displays the group.
StrikeOut	Specifies whether the group's caption should appear in strikeout.
ToolTip	Specifies the group's tooltip.
Underline	Specifies whether the group's caption appears as underlined.
UserData	Specifies an extra data.

method Group.AddItem (Caption as String, [Image as Variant])

Adds a new item to the group object.

Type	Description
Caption as String	A string expression that indicates the item's caption. The Caption may includes built-in HTML format if the CaptionFormat property is exHTML.
Image as Variant	A long expression that indicates the item's icon
Return	Description
Item	An Item object being added.

Use the AddItem method to add new items to group. Use the [Add](#) method property to add new groups to the control. The [AddItem](#) event is fired each time when a new item is added to group. Use the [BeginUpdate](#) and [EndUpdate](#) methods to maintain performance while adding new groups or new items. The [Group](#) property specifies the group that owns the item. Use the [Image](#) property to specify the item's picture. Use the [Indent](#) property to indent the item. Use the [Caption](#) property to specify the caption of the item. Use the [CaptionFormat](#) property to allow built-in HTML tags to the group's caption.

The following VB sample adds two groups and two items to each group:

```
With ListBar1
    .BeginUpdate
    With .Groups
        With .Add("Group 1")
            .AddItem "Item 1"
            .AddItem "Item 2"
        End With
        With .Add("Group 2")
            .AddItem "Item 1"
            .AddItem "Item 2"
        End With
    End With
    .EndUpdate
End With
```

The following C++ sample adds two groups and two items to each group:

```
#include "Item.h"
```

```

#include "Group.h"
#include "Groups.h"
COleVariant vtMissing; V_VT( &vtMissing ) = VT_ERROR;
m_listbar.BeginUpdate();
CGroups groups = m_listbar.GetGroups();
CGroup group1 = groups.Add( "Group 1" );
group1.AddItem( "Item 1", vtMissing );
group1.AddItem( "Item 2", vtMissing );
CGroup group2 = groups.Add( "Group 2" );
group2.AddItem( "Item 1", vtMissing );
group2.AddItem( "Item 2", vtMissing );
m_listbar.EndUpdate();

```

The following VB.NET sample adds two groups and two items to each group:

```

With AxListBar1
    .BeginUpdate()
    With .Groups
        With .Add("Group 1")
            .AddItem("Item 1")
            .AddItem("Item 2")
        End With
        With .Add("Group 2")
            .AddItem("Item 1")
            .AddItem("Item 2")
        End With
    End With
    .EndUpdate()
End With

```

The following C# sample adds two groups and two items to each group:

```

axListBar1.BeginUpdate();
EXLISTBARLib.Group group1 = axListBar1.Groups.Add("Group 1");
group1.AddItem("Item 1", null);
group1.AddItem("Item 2", null);
EXLISTBARLib.Group group2 = axListBar1.Groups.Add("Group 2");
group2.AddItem("Item 1", null);

```

```
group2.AddItem("Item 2", null);  
axListBar1.EndUpdate();
```

The following VFP sample adds two groups and two items to each group:

```
With thisform.ListBar1  
  .BeginUpdate()  
  With .Groups  
    With .Add("Group 1")  
      .AddItem("Item 1")  
      .AddItem("Item 2")  
    EndWith  
    With .Add("Group 2")  
      .AddItem("Item 1")  
      .AddItem("Item 2")  
    EndWith  
  EndWith  
  .EndUpdate()  
EndWith
```

property Group.Alignment as AlignmentEnum

Retrieves or sets a value that indicates the caption's alignment.

Type	Description
AlignmentEnum	An AlignmentEnum expression that indicates the group caption's alignment.

By default, the Alignment property is exCenter. The Alignment property doesn't align items in the group. Use the [Alignment](#) property to align an item. If you want to align the entire list of items, you can handle the [AddItem](#) event where you can change the item's alignment each time when a new item is added to group.

property Group.AutoScroll as Boolean

Specifies whether the scroll buttons are automatically added.

Type	Description
Boolean	A boolean expression that indicates whether the control adds scroll buttons if the items list doesn't fit the the group's client area.

By default, the AllowScroll property is True. Use the AllowScroll property to specify whether the group is scrolled smoothly until all items are visible when the user clicks the group's caption. Use the [DelayScroll](#) property to specify the delay used when the user selects a group. Use the AllowScroll property on False, when the group hosts an ActiveX control or another window.

property Group.BackColor as Color

Retrieves or sets the group's background color.

Type	Description
Color	A color expression that indicates the background color of the group's caption. 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 BackColor property to specify the group's caption's background color. Use the [BackColor2](#) property to specify the color at the ending boundary line of the gradient group's caption. Use the [BackColorList](#) property to specify the background color for group's list. Use the [BackColorGroup](#) property to specify a default background color for groups. Use the <bgcolor> built in HTML tag in the [Caption](#) property to define portions of text using a specified background color.



In VB.NET or C# you require the following functions until the .NET framework will provide:

You can use the following VB.NET function:

```
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
```

You can use the following C# function:

```
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 VB sample changes the group's background color:

```
With ListBar1.Groups(0)  
    .BackColor = vbBlue  
End With
```

The following C++ sample changes the group's background color:

```
m_listbar.GetGroups().GetItem( COleVariant( long(0) ) ).SetBackColor( RGB(0,0,255) );
```

The following VB.NET sample changes the group's background color:

```
With AxListBar1.Groups(0)  
    .BackColor = ToUInt32(Color.Blue)  
End With
```

The following C# sample changes the group's background color:

```
axListBar1.Groups[0].BackColor = ToUInt32(Color.Blue);
```

The following VFP sample changes the group's background color:

```
With thisform.ListBar1.Groups(0)  
    .BackColor = RGB(0,0,255)  
EndWith
```

property Group.BackColor2 as Color

Specifies the color at the ending boundary line of the gradient group's caption.

Type	Description
Color	A color expression that specifies the color at the ending boundary line of the gradient group's caption.

Use the BackColor2 property to specify the second background color when painting its background in gradient. Use the [BackColor](#) and [ForeColor](#) properties to specify the item's background and foreground colors. Use the [BackColorList](#) property to specify the default background color for the group's list. Use the [BackColor](#) property to specify the control's background color.



In VB.NET or C# you require the following functions until the .NET framework will provide:

You can use the following VB.NET function:

```
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
```

You can use the following C# function:

```
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);
}
```


property Group.BackColorList as Color

Retrieves or sets a value that indicates the background color of the list when the group is active.

Type	Description
Color	A color expression that indicates the background color of the group's list.

The BackColorList property has effect only when the group is selected (active). Use the [BackColor](#) property to specify the background color for group's caption. Use the [BackColor](#) property to specify the control's background color. Use the [BackColorGroup](#) property to specify a default background color for groups. Use the <bgcolor> built in HTML tag in the [Caption](#) property to define portions of text using a specified background color.

property Group.Bold as Boolean

Specifies whether the group's caption should appear in bold.

Type	Description
Boolean	A boolean expression that specifies whether the the group's caption should appear in bold.

Use the Bold, [Italic](#), [Underline](#) and [StrikeOut](#) properties to apply different font attributes to the group. Use the [Caption](#) property to display different parts of the caption using HTML format. Use the [Font](#) property to specify the control's font. Use the [Bold](#) property to bold an item. Use the HTML tag to specify parts of group's caption that should appear in bold, if [CaptionFormat](#) property is exHTML.

The following VB sample bolds all groups:

```
Private Sub ListBar1_AddGroup(ByVal Group As EXLISTBARLibCtl.IGroup)
    With Group
        .Bold = True
    End With
End Sub
```

The following C++ sample bolds all groups:

```
void OnAddGroupListbar1(LPDISPATCH Group)
{
    CGroup group( Group ); group.m_bAutoRelease = FALSE;
    group.SetBold( TRUE );
}
```

The following VB.NET sample bolds all groups:

```
Private Sub AxListBar1_AddGroup(ByVal sender As System.Object, ByVal e As
AxEXLISTBARLib._IListBarEvents_AddGroupEvent) Handles AxListBar1.AddGroup
    With e.group
        .Bold = True
    End With
End Sub
```

The following C# sample bolds all groups:

```
private void axListBar1_AddGroup(object sender,  
AxEXLISTBARLib._IListBarEvents_AddGroupEvent e)  
{  
    e.group.Bold = true;  
}
```

The following VFP sample bolds all groups:

```
*** ActiveX Control Event ***  
LPARAMETERS group  
  
with group  
    .Bold = .t.  
endwith
```

property Group.Caption as String

Specifies the group's caption.

Type	Description
String	A string expression that indicates the group's caption.

You can use the [Add](#) method to specify the group's caption, when the group is added to groups collection. Use the [CaptionFormat](#) property to allow built-in HTML tags in the group's caption. The control fires the [SelectGroup](#) event when the user clicks the caption of the group. Use the [Bold](#), [Italic](#), [Underline](#) or [StrikeOut](#) property to define the font for caption of the group. Use the **** HTML tag to insert icons inside the item's caption, if the [CaptionFormat](#) property is exHTML. For instance, the "some image **1** other image **2** rest of text", displays combined text and icons in the item's caption. Use the [Images](#) method to load icons at runtime.

property Group.CaptionFormat as CaptionFormatEnum

Specifies how the group's caption is displayed.

Type	Description
CaptionFormatEnum	A CaptionFormatEnum expression that indicates whether the control uses built-in HTML tags.

By default, the CaptionFormat property is exText. Use the CaptionFormat property to allow built-in HTML tags to the group's caption. Use the [Caption](#) property to access the group's caption. Use the [AddItem](#) method to add a new item to the group. The Caption property supports the following built-in HTML tags:

- ** bold **
- **<u> underline </u>**
- **<s> strikeout </s>**
- **<i> italic </i>**
- **<fgcolor = FF0000> fgcolor </fgcolor>**
- **<bgcolor = FF0000> bgcolor </bgcolor>**
- **
** breaks a line.
- **<solidline>** draws a solid line
- **<dotline>** draws a dotted line
- **<upline>** draws the line to the top of the text line
- **<r>** aligns the rest of the text line to the right side
- **number** inserts an icon inside the item's caption. The number indicates the index of the icon being inserted.
- **key[:width]** inserts a custom size picture being 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.
- **text ** displays portions of text with a different font and/or different size. For instance, the **bit** draws the bit text using the Tahoma font, on size 12 pt. If the name of the font is missing, and instead size is present, the current font is used with a different size. For instance, **bit** displays the bit text using the current font, but with a different size.

Newer HTML format supports subscript and superscript like follows:

- **<off offset> ... </off>** defines the vertical offset to display the text/element. The offset parameter defines the offset to display the element. This tag is inheritable, so the offset is keep while the associated **</off>** tag is found. You can use the **<off offset>** HTML tag in combination with the **** to define a smaller or a larger font

to be displayed. For instance: *"Text with <off 6>subscript"* displays the text such as: Text with subscript The *"Text with <off -6>superscript"* displays the text such as: Text with subscript

Also, newer HTML format supports decorative text like follows:

- **<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 FFFFFFFF;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 adds an HTML item:

```
With ListBar1.Groups(0)
    With .AddItem("This is an <fgcolor=000080> <b>HTML</b> </fgcolor> item.")
        .CaptionFormat = exHTML
    End With
End With
```

The following C++ sample adds an HTML item:

```
COleVariant vtMissing; V_VT( &vtMissing ) = VT_ERROR;
CGroup group = m_listbar.GetGroups().GetItem( COleVariant( long(0) ) );
CItem item = group.AddItem( "This is an <fgcolor=000080> <b>HTML</b> </fgcolor> item.", vtMissing );
item.SetCaptionFormat( 1 /*exHTML*/ );
```

The following VB.NET sample adds an HTML item:

```
With AxListBar1.Groups(0)
    With .AddItem("This is an <fgcolor=000080> <b>HTML</b> </fgcolor> item.")
        .CaptionFormat = EXLISTBARLib.CaptionFormatEnum.exHTML
    End With
End With
```

The following C# sample adds an HTML item:

```
EXLISTBARLib.Item item = axListBar1.Groups[0].AddItem("This is an <fgcolor=000080> <b>HTML</b> </fgcolor> item.", null);
item.CaptionFormat = EXLISTBARLib.CaptionFormatEnum.exHTML;
```

The following VFP sample adds an HTML item:

```
With thisform.ListBar1.Groups(0)
    With .AddItem("This is an <fgcolor=000080> <b>HTML</b> </fgcolor> item.")
        .CaptionFormat = 1 && exHTML
    End With
End With
```


EndWith
EndWith

method Group.Clear ()

Clears the items collection.

Type	Description
------	-------------

Use the Clear method to clear the Items collection. Use the [RemoveItem](#) method to remove a specific item. Use the the [Visible](#) property to hide an item. The control fires the [RemoveItem](#) event when an item is removed. Use the [Count](#) property to count the items in the group. Use the [Item](#) property to access an item giving its index or by its caption. Use the [ItemByPos](#) property to get the item giving its position. Use the [ItemHeight](#) property to specify the height for all items in the group.

property Group.Count as Long

Gets the count of the items.

Type	Description
Long	A long expression that indicates the count of items in the group.

The Count property counts the items in the group. Use the [Item](#) property to access an item giving its index or by its caption. Use the [ItemByPos](#) property to get the item giving its position. Use the [ItemHeight](#) property to specify the height for all items in the group. Use the [Caption](#) property to specify the caption of the item. Use the [Visible](#) property to specify whether an item is visible or hidden.

The following VB sample enumerates the items in the group:

```
With ListBar1.Groups(0)
    Dim i As Long
    For i = 0 To .Count - 1
        Debug.Print .Item(i).Caption
    Next
End With
```

The following VB sample enumerates the items in the group, as they are displayed:

```
With ListBar1.Groups(0)
    Dim i As Long
    For i = 0 To .Count - 1
        Dim it As EXLISTBARLibCtl.Item
        Set it = .ItemByPos(i)
        If it.Visible Then
            Debug.Print it.Caption
        End If
    Next
End With
```

The following C++ sample enumerates the items in the group:

```
CGroup group = m_listbar.GetGroups().GetItem( COleVariant( long(0) ) );
for ( long i = 0; i < group.GetCount(); i++ )
```

```
{
    CItem item = group.GetItem( COleVariant( long(i) ) );
    OutputDebugString( item.GetCaption() );
}
```

The following C++ sample enumerates the items in the group, as they are displayed:

```
CGroup group = m_listbar.GetGroups().GetItem( COleVariant( long(0) ) );
for ( long i = 0; i < group.GetCount(); i++ )
{
    CItem item = group.GetItemByPos( i );
    if ( item.GetVisible() )
        OutputDebugString( item.GetCaption() );
}
```

The following VB.NET sample enumerates the items in the group:

```
With AxListBar1.Groups(0)
    Dim i As Integer
    For i = 0 To .Count - 1
        Debug.WriteLine(.Item(i).Caption())
    Next
End With
```

The following VB.NET sample enumerates the items in the group, as they are displayed:

```
With AxListBar1.Groups(0)
    Dim i As Long
    For i = 0 To .Count - 1
        Dim it As EXLISTBARLib.Item = .ItemByPos(i)
        If it.Visible Then
            Debug.WriteLine(it.Caption)
        End If
    Next
End With
```

The following C# sample enumerates the items in the group:

```
EXLISTBARLib.Group group = axListBar1.Groups[0];
```

```
for (int i = 0; i < group.Count; i++)  
    System.Diagnostics.Debug.WriteLine(group[i].Caption);
```

The following C# sample enumerates the items in the group, as they are displayed:

```
EXLISTBARLib.Group group = axListBar1.Groups[0];  
for (int i = 0; i < group.Count; i++)  
{  
    EXLISTBARLib.Item item = group.get_ItemByPos(i);  
    if ( item.Visible )  
        System.Diagnostics.Debug.WriteLine(item.Caption);  
}
```

The following VFP sample enumerates the items in the group:

```
With thisform.ListBar1.Groups(0)  
    local i  
    For i = 0 To .Count - 1  
        wait window nowait .Item(i).Caption  
    Next  
EndWith
```

The following VFP sample enumerates the items in the group, as they are displayed:

```
With thisform.ListBar1.Groups(0)  
    local i  
    For i = 0 To .Count - 1  
        local it  
        it = .ItemByPos(i)  
        If it.Visible Then  
            wait window nowait it.Caption  
        EndIf  
    Next  
EndWith
```

method Group.EnsureVisibleItem (Item as Long)

Ensures that the item fits the group's client area.

Type	Description
Item as Long	A long expression that indicates the index of item.

The EnsureVisibleItem method scrolls the group's list to make sure that the item fits the group's client area.

property Group.ForeColor as Color

Specifies the group's foreground color.

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

Use the ForeColor property to specify the group's caption foreground color. Use the [ForeColorList](#) property to specify the foreground color of the group's list. Use the [ForeColorGroup](#) property to specify the default foreground color. Use the <fgcolor> built in HTML tag in the [Caption](#) property to define portions of text using a specified foreground color.

In VB.NET or C# you require the following functions until the .NET framework will provide:

You can use the following VB.NET function:

```
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
```

You can use the following C# function:

```
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 VB sample changes the group's foreground color:

```
With ListBar1.Groups(0)
    .ForeColor = vbBlue
```

End With

The following C++ sample changes the group's foreground color:

```
m_listbar.GetGroups().GetItem( COleVariant( long(0) ) ).SetForeColor( RGB(0,0,255) );
```

The following VB.NET sample changes the group's foreground color:

```
With AxListBar1.Groups(0)  
    .ForeColor = ToUInt32(Color.Blue)  
End With
```

The following C# sample changes the group's foreground color:

```
axListBar1.Groups[0].ForeColor = ToUInt32(Color.Blue);
```

The following VFP sample changes the group's foreground color:

```
With thisform.ListBar1.Groups(0)  
    .ForeColor = RGB(0,0,255)  
EndWith
```


property Group.ForeColorList as Color

Retrieves or sets a value that indicates the foreground color of the group's list when it is active.

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

Use the ForeColorList property to specify the foreground color for group's list. Use the [ForeColor](#) property to specify the foreground color of the group's caption. Use the [ForeColor](#) property to specify the item's foreground color. Use the [ForeColor](#) property to specify the control's foreground color. Use the <fgcolor> built in HTML tag in the [Caption](#) property to define portions of text using a specified foreground color. Use the ForeColorList property to specify the foreground color for group's list. Use the [ForeColor](#) property to specify the foreground color of the group's caption. Use the [ForeColor](#) property to specify the item's foreground color. Use the [ForeColor](#) property to specify the control's foreground color. Use the <fgcolor> built in HTML tag in the [Caption](#) property to define portions of text using a specified foreground color.

In VB.NET or C# you require the following functions until the .NET framework will provide:

You can use the following VB.NET function:

```
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
```

You can use the following C# function:

```
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);
}
```


property Group.Image as Variant

Specifies the index of the group's icon.

Type	Description
Variant	A long expression that indicates the index of icon being used, a string expression that indicates the base64 encoded string that holds a picture object, a Picture object. Use the eximages tool to save your picture as base64 encoded format.

Use the Image property to assign an icon or a picture to the group. Use the [Images](#) and [Replacelcon](#) methods to update the images list collection, at runtime. Use the [Image](#) property to assign a picture to an item. The [GroupHeight](#) property specifies the height of the caption for all groups, in pixels. Use the **** HTML tag to insert icons inside the iteml's caption, if the [CaptionFormat](#) property is exHTML.

The following VB sample loads a collection of icons from a BASE64 encoded string:

```
With ListBar1
  Dim s As String
  s =
"gBJJgBAIEAAGAEGCAAhb/hz/EIAh8Tf5CJo2AEZjQAjEZFEaIEaEEaAIAkcbk0oIUrlktl0vmExml

  s = s +
"FphZDEJtT1zp7bd1XasiLB8ld4o8kCRJIACSpRfNKWXd1uH+eaHn8P55necB/kmf5+A+eBnr

  .BeginUpdate
  .Images s
  With .Groups
    With .Add("Group 1")
      .Image = 1
    End With
  End With
  .EndUpdate
End With
```

The following C++ sample loads a collection of icons from a BASE64 encoded string:

```
#include "Item.h"
```

```

#include "Group.h"
#include "Groups.h"
COleVariant vtMissing; V_VT( &vtMissing ) = VT_ERROR;
CString
s("gBJJgBAIEAAGAEGCAAhb/hz/EIAh8Tf5CJo2AEZjQAjEZFEaIEaEEaAIAkcbk0oIUrlktl0vmExr

s = s +
"FphZDEJtT1zp7bd1XasiLB8ld4o8kCRJIACSpRfNKWXd1uH+eaHn8P55necB/kmf5+A+eBnr

m_listbar.BeginUpdate();
CGroups groups = m_listbar.GetGroups();
m_listbar.Images(COleVariant(s));
CGroup group = groups.Add( "Group 1" );
group.SetImage( COleVariant( (long)1 ) );
m_listbar.EndUpdate();

```

The following VB.NET sample loads a collection of icons from a BASE64 encoded string:

```

With AxListBar1
    Dim s As String =
"gBJJgBAIEAAGAEGCAAhb/hz/EIAh8Tf5CJo2AEZjQAjEZFEaIEaEEaAIAkcbk0oIUrlktl0vmExml

    s = s +
"FphZDEJtT1zp7bd1XasiLB8ld4o8kCRJIACSpRfNKWXd1uH+eaHn8P55necB/kmf5+A+eBnr

    .BeginUpdate()
    .Images(s)
    With .Groups
        With .Add("Group 1")
            .Image = 1
        End With
    End With
    .EndUpdate()
End With

```

The following C# sample loads a collection of icons from a BASE64 encoded string:

```

String s =

```

```
"gBJJgBAIEAAGAEGCAAhb/hz/EIAh8Tf5CJo2AEZjQAjEZFEaIEaEEaAIAkcbk0oIUrlktl0vmExml
```

```
s = s +
```

```
"FphZDEJtT1zp7bd1XasiLB8ld4o8kCRJIACSpRfNKWXd1uH+eaHn8P55necB/kmf5+A+eBnr
```

```
axListBar1.BeginUpdate();
```

```
axListBar1.Images(s);
```

```
EXLISTBARLib.Group group = axListBar1.Groups.Add("Group 1");
```

```
group.Image = 1;
```

```
axListBar1.EndUpdate();
```

The following VFP sample loads a collection of icons from a BASE64 encoded string:

```
With thisform.ListBar1
```

```
    local s
```

```
    .BeginUpdate()
```

```
    s =
```

```
"gBJJgBAIEAAJAEGCAAhb/hz/EIAh8Tf5CJo2AEZjQAjEZFEaIEaGEaAIAEEbjMjlErlktl0vmExmU
```

```
    s = s +
```

```
"Fw2HxGJxWLxmNx2PyGRyWTymVy2XzGZzWbzmdz2f0Gh0Wj0ml02npqAQEZ1WojWq1b.
```

```
    s = s +
```

```
"Gjb8wQ4L1us7zeQUmbqP29z4uu3rWv+ +jVwo7kKus3MNRA1cOQk6SgRPEkMwu3EXRY9'
```

```
    s = s +
```

```
"A/koxFKE6Q3O6WT9M8MSHQ0MuXQLpSXLE3xZJUwJy8bW0W+UPT7DEnUIMEa0xHdMC
```

```
    s = s +
```

```
"pdLzXZfsNX8+qfUEs2Bxg5lJwerN7Jld7LWlZ2Atc0qLB8lmKo8kCRJIACSo2QGO11hOJM2hx
```

```
    s = s + "+YJ/7YPaHnGiaAg="
```

```
    With .Groups
```

```
        With .Add("Group 1")
```

```
            .Image = 1
```

```
        EndWith
```

```
    EndWith
```

```
.EndUpdate()
```

```
EndWith
```

The following Template sample loads a collection of icons from a BASE64 encoded string:

```
BeginUpdate
```

```
Images("gBJJgBAIEAAJAEGCAAhb/hz/EIAh8Tf5CJo2AEZjQAjEZFEaIEaGEaAIAEEbjMjIErIktlOv
```

```
Groups
```

```
{  
    "Group 1"  
    {  
        Image = 1  
    }  
}
```

```
EndUpdate
```

property Group.IndentHeaderBottom as Long

Specifies the number of pixels to indent the group's header from the bottom part.

Type	Description
Long	A long expression that specifies the number of pixels to indent the groups' header from the top side.

By default, the IndentHeaderRight property is 0. The IndentHeaderLeft property has effect only for group's [caption](#), [image](#) and it does not affect the group's header background appearance. Use the [BackColorGroup](#) property to define a new background appearance using EBN files. Use the [CP](#) option (copy option) of the EBN files to define the way EBN file is arranged on the object. Use the [IndentHeaderLeft](#) property to indent the group's header from left side. Use the [IndentHeaderRight](#) property to indent the group's header from right side. Use the [IndentHeaderTop](#) property to indent the group's header from top side. Use the IndentHeaderBottom property to indent the group's header from bottom side.

The following screen shot shows the captions being displayed when indent properties has been used:



The following screen shot shows the captions being displayed when indent properties has not been used:



property Group.IndentHeaderLeft as Long

Specifies the number of pixels to indent the group's header from the left part.

Type	Description
Long	A long expression that specifies the number of pixels to indent the groups' header from the left side.

By default, the IndentHeaderLeft property is 0. The IndentHeaderLeft property has effect only for group's [caption](#), [image](#) and it does not affect the group's header background appearance. Use the [BackColorGroup](#) property to define a new background appearance using EBN files. Use the [CP](#) option (copy option) of the EBN files to define the way EBN file is arranged on the object. Use the IndentHeaderLeft property to indent the group's header from left side. Use the [IndentHeaderRight](#) property to indent the group's header from right side. Use the [IndentHeaderTop](#) property to indent the group's header from top side. Use the [IndentHeaderBottom](#) property to indent the group's header from bottom side.

The following screen shot shows the captions being displayed when indent properties has been used:



The following screen shot shows the captions being displayed when indent properties has not been used:



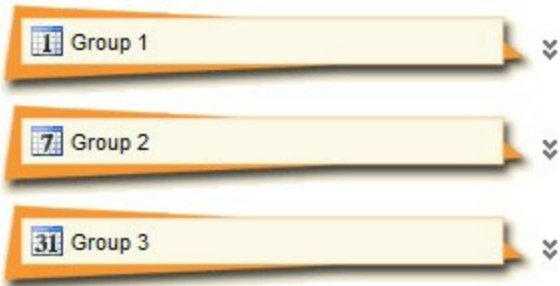
property Group.IndentHeaderRight as Long

Specifies the number of pixels to indent the group's header from the right part.

Type	Description
Long	A long expression that specifies the number of pixels to indent the groups' header from the right side.

By default, the IndentHeaderRight property is 0. The IndentHeaderLeft property has effect only for group's [caption](#), [image](#) and it does not affect the group's header background appearance. Use the [BackColorGroup](#) property to define a new background appearance using EBN files. Use the [CP](#) option (copy option) of the EBN files to define the way EBN file is arranged on the object. Use the [IndentHeaderLeft](#) property to indent the group's header from left side. Use the IndentHeaderRight property to indent the group's header from right side. Use the [IndentHeaderTop](#) property to indent the group's header from top side. Use the [IndentHeaderBottom](#) property to indent the group's header from bottom side.

The following screen shot shows the captions being displayed when indent properties has been used:



The following screen shot shows the captions being displayed when indent properties has not been used:



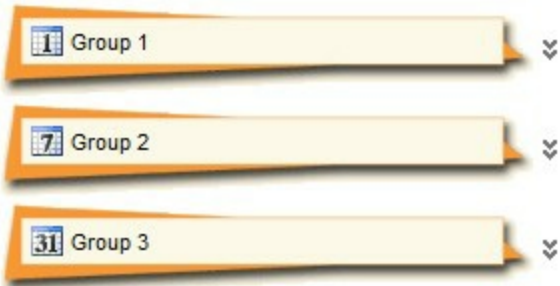
property Group.IndentHeaderTop as Long

Specifies the number of pixels to indent the group's header from the top part.

Type	Description
Long	A long expression that specifies the number of pixels to indent the groups' header from the top side.

By default, the IndentHeaderRight property is 0. The IndentHeaderLeft property has effect only for group's [caption](#), [image](#) and it does not affect the group's header background appearance. Use the [BackColorGroup](#) property to define a new background appearance using EBN files. Use the [CP](#) option (copy option) of the EBN files to define the way EBN file is arranged on the object. Use the [IndentHeaderLeft](#) property to indent the group's header from left side. Use the [IndentHeaderRight](#) property to indent the group's header from right side. Use the IndentHeaderTop property to indent the group's header from top side. Use the [IndentHeaderBottom](#) property to indent the group's header from bottom side.

The following screen shot shows the captions being displayed when indent properties has been used:



The following screen shot shows the captions being displayed when indent properties has not been used:



property Group.Index as Long

Retrieves the index of the object into the Groups collection..

Type	Description
Long	A long expression that indicates the group's index into the groups collection.

Use the Index property to identify a Group object into the groups collection. Use the [Caption](#) property to specify the caption of the group. Use the [Position](#) property to specify the group's position. Use the [Count](#) property to count the groups in the control. Use the [Item](#) property to access a group by its index or by its caption. Use the [ItemByPos](#) property to retrieve the group by position.

property Group.Italic as Boolean

Specifies whether the group's caption should appear in italic.

Type	Description
Boolean	A boolean expression that specifies whether the group's caption should appear in italic.

Use the [Bold](#), [Italic](#), [Underline](#) and [StrikeOut](#) properties to apply different font attributes to the group. Use the [Caption](#) property to display different parts of the caption using HTML format. Use the [Font](#) property to specify the control's font. Use the [Italic](#) property to make an item appear in italic. Use the <i> HTML tag to specify parts of group's caption that should appear in italic, if [CaptionFormat](#) property is exHTML.

The following VB sample makes all groups appear in italic:

```
Private Sub ListBar1_AddGroup(ByVal Group As EXLISTBARLibCtl.IGroup)
    With Group
        .Italic = True
    End With
End Sub
```

The following C++ sample makes all groups appear in italic:

```
void OnAddGroupListbar1(LPDISPATCH Group)
{
    CGroup group( Group ); group.m_bAutoRelease = FALSE;
    group.SetBold( TRUE );
}
```

The following VB.NET sample makes all groups appear in italic:

```
Private Sub AxListBar1_AddGroup(ByVal sender As System.Object, ByVal e As
AxEXLISTBARLib._IListBarEvents_AddGroupEvent) Handles AxListBar1.AddGroup
    With e.group
        .Italic = True
    End With
End Sub
```

The following C# sample makes all groups appear in italic:

```
private void axListBar1_AddGroup(object sender,  
AxEXLISTBARLib._IListBarEvents_AddGroupEvent e)  
{  
    e.group.Italic = true;  
}
```

The following VFP sample makes all groups appear in italic:

```
*** ActiveX Control Event ***  
LPARAMETERS group  
  
with group  
    .Italic = .t.  
endwith
```

property Group.Item (Index as Variant) as Item

Returns a specific Item from the collection.

Type	Description
Index as Variant	A long expression that indicates the index of the item, or a string expression that indicates the item's caption
Item	An Item object being accessed.

Use the Item property to access items of the group. Use the [Count](#) property to get the number of items in the group. Use the [ItemByPos](#) to access items by position. Use the [ItemHeight](#) property to specify the height for all items in the group. Use the [Caption](#) property to specify the caption of the item. Use the [Visible](#) property to specify whether an item is visible or hidden.

The following VB sample enumerates the items in the group:

```
With ListBar1.Groups(0)
    Dim i As Long
    For i = 0 To .Count - 1
        Debug.Print .Item(i).Caption
    Next
End With
```

The following C++ sample enumerates the items in the group:

```
CGroup group = m_listbar.GetGroups().GetItem( COleVariant( long(0) ) );
for ( long i = 0; i < group.GetCount(); i++ )
{
    CItem item = group.GetItem( COleVariant( long(i) ) );
    OutputDebugString( item.GetCaption() );
}
```

The following VB.NET sample enumerates the items in the group:

```
With AxListBar1.Groups(0)
    Dim i As Integer
    For i = 0 To .Count - 1
        Debug.WriteLine(.Item(i).Caption())
    Next
```

End With

The following C# sample enumerates the items in the group:

```
EXLISTBARLib.Group group = axListBar1.Groups[0];  
for (int i = 0; i < group.Count; i++)  
    System.Diagnostics.Debug.WriteLine(group[i].Caption);
```

The following VFP sample enumerates the items in the group:

```
With thisform.ListBar1.Groups(0)  
    local i  
    For i = 0 To .Count - 1  
        wait window nowait .Item(i).Caption  
    Next  
EndWith
```

property Group.ItemByPos (Position as Long) as Item

Gets the item given its position.

Type	Description
Position as Long	A long expression that indicates the item's position.
Item	An Item object being accessed.

Use the ItemByPos property to access the items in the group by position. The Position property is 0 based. The [Count](#) property counts the items in the group. Use the [ItemHeight](#) property to specify the height for all items in the group. Use the [Caption](#) property to specify the caption of the item. Use the [Visible](#) property to specify whether an item is visible or hidden.

The following VB sample enumerates the items in the group, as they are displayed:

```
With ListBar1.Groups(0)
    Dim i As Long
    For i = 0 To .Count - 1
        Dim it As EXLISTBARLibCtl.Item
        Set it = .ItemByPos(i)
        If it.Visible Then
            Debug.Print it.Caption
        End If
    Next
End With
```

The following C++ sample enumerates the items in the group, as they are displayed:

```
CGroup group = m_listbar.GetGroups().GetItem( COleVariant( long(0) ) );
for ( long i = 0; i < group.GetCount(); i++ )
{
    CItem item = group.GetItemByPos( i );
    if ( item.GetVisible() )
        OutputDebugString( item.GetCaption() );
}
```

The following VB.NET sample enumerates the items in the group, as they are displayed:

```
With AxListBar1.Groups(0)
```



```

Dim i As Long
For i = 0 To .Count - 1
    Dim it As EXLISTBARLib.Item = .ItemByPos(i)
    If it.Visible Then
        Debug.WriteLine(it.Caption)
    End If
Next
End With

```

The following C# sample enumerates the items in the group, as they are displayed:

```

EXLISTBARLib.Group group = axListBar1.Groups[0];
for (int i = 0; i < group.Count; i++)
{
    EXLISTBARLib.Item item = group.get_ItemByPos(i);
    if ( item.Visible )
        System.Diagnostics.Debug.WriteLine(item.Caption);
}

```

The following VFP sample enumerates the items in the group, as they are displayed:

```

With thisform.ListBar1.Groups(0)
    local i
    For i = 0 To .Count - 1
        local it
        it = .ItemByPos(i)
        If it.Visible Then
            wait window nowait it.Caption
        EndIf
    Next
EndWith

```

property Group.ItemHeight as Long

Retrieves or sets the item's height.

Type	Description
Long	A long expression that indicates the item's height.

Use the ItemHeight and [ItemWidth](#) properties to specify the size for the item in the group. By default, the ItemHeight property is 24 pixels. The [GroupHeight](#) property specifies the height of the caption for all groups, in pixels. Use the [Font](#) property to specify the control's font. Use the [Caption](#) property to specify the caption of the item. Use the [SmallIcons](#) property to specify the size of the icons being displayed. The [AddGroup](#) event notifies the control that a new groups is added.

property Group.ItemWidth as Long

Retrieves or sets the item's width. -1 if it's the group's client width.

Type	Description
Long	A long expression that indicates the item's width.

Use the [ItemHeight](#) and ItemWidth properties to specify the size for the item in the group. If the ItemWidth property is grater than zero, the control auto arranges the items in the group to fit its client area. If the ItemWidth property is less than zero only one item is displayed into a line.

property Group.Picture as IPictureDisp

Retrieves or sets a graphic to be displayed in the group's list.

Type	Description
IPictureDisp	A Picture property that specifies the group's picture.

The Picture property specifies the group's background picture. Use the [PictureDisplay](#) property to determine how the picture is arranged on the group's background. Use the [BackColorList](#) property to specify the background color for the group's list. Use the [BackColor](#) property to specify the control's background color. Use the [Picture](#) property to display a picture on the control's background list.



property Group.PictureDisplay as PictureDisplayEnum

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

Type	Description
PictureDisplayEnum	A PictureDisplayEnum expression that indicates the way how the group's Picture is arranged.

By default, the group's PictureDisplay property is exTile. Use the PictureDisplay property to determine how the picture is arranged on the group's background. The [Picture](#) property specifies the group's background picture. Use the [BackColorList](#) property to specify the background color for the group's list. Use the [BackColor](#) property to specify the control's background color. Use the [Picture](#) property to display a picture on the control's background list.

property Group.Position as Long

Specifies the group's position.

Type	Description
Long	A long expression that indicates the group's position.

Use the Position property to arrange groups. Use the [Item](#) property to access the group giving its index or caption. Use the [Caption](#) property to get the group's caption. Use the [Index](#) property to identify a group. Use the [ItemByPos](#) property to access groups by position. The Position property is zero based. For instance, if the Position property is zero, it means first visible group. Use the [Visible](#) property to specify whether an item is visible or hidden.

The following VB sample enumerates the groups in the control, as they are displayed:

```
With ListBar1.Groups
    Dim i As Long
    For i = 0 To .Count - 1
        With .ItemByPos(i)
            Debug.Print (.Caption)
        End With
    Next
End With
```

The following C++ sample enumerates the groups in the control, as they are displayed:

```
CGroups groups = m_listbar.GetGroups();
for ( long i = 0; i < groups.GetCount(); i++ )
{
    CGroup group( groups.GetItemByPos( i ) );
    OutputDebugString( group.GetCaption() );
}
```

The following VB.NET sample enumerates the groups in the control, as they are displayed:

```
With AxListBar1.Groups
    Dim i As Integer
    For i = 0 To .Count - 1
        With .ItemByPos(i)
```

```
        Debug.WriteLine(.Caption)
    End With
Next
End With
```

The following C# sample enumerates the groups in the control, as they are displayed:

```
for (int i = 0; i < axListBar1.Groups.Count; i++)
{
    EXLISTBARLib.Group g = axListBar1.Groups.get_ItemByPos(i);
    System.Diagnostics.Debug.WriteLine(g.Caption);
}
```

The following VFP sample enumerates the groups in the control, as they are displayed:

```
With thisform.ListBar1.Groups
    local i
    For i = 0 To .Count - 1
        With .ItemByPos(i)
            wait window nowait .Caption
        EndWith
    Next
EndWith
```

method Group.RemoveItem (Index as Variant)

Removes an item given its index or given its caption.

Type	Description
Index as Variant	A long expression that indicates the index of item being removed, or a string expression that indicates the item's caption being removed

Use the RemoveItem to remove items in the group. When an item is removed the [RemoveItem](#) event is fired. Use the [Clear](#) method to clear the entire collection of items in the group. Use the [Visible](#) property to specify whether the item is visible or hidden. Use the [ItemByPos](#) property to get the item giving its position. Use the [ItemHeight](#) property to specify the height for all items in the group. Use the [Count](#) property to count the items in the group. Use the [Item](#) property to access an item giving its index or by its caption.

property Group.SelectItem as Long

Specifies the index of item that's selected.

Type	Description
Long	A long expression that indicates the index of selected item in the group.

Use the SelectItem property to select items. The [SelectItem](#) event is fired when an item is selected, or unselected. Use the [Count](#) property to count items in the group. Use the [Item](#) property to access an item giving its index or by its caption. Use the [ItemByPos](#) property to get the item giving its position. Use the [ItemHeight](#) property to specify the height for all items in the group. Use the [Caption](#) property to specify the caption of the item. Use the [Visible](#) property to specify whether an item is visible or hidden.

property Group.Shortcut as String

Specifies the name of the shortcut which displays the group.

Type	Description
String	A HTML expression that indicates the caption of the shortcut.

The Group objects with the same Shortcut property belongs to the same set, and displays the Shortcut caption in the control's shortcut bar. The [ShowShortcutBar](#) property specifies whether the control's shortcut bar is visible or hidden. By default, the Shortcut property is empty, so all Group in the Groups collection belongs to the same set. The shortcut bar displays the first icon in the HTML caption, if found, or it displays a custom size picture if specified using the the [ShortcutPicture](#) property. If the Shortcut has associated a custom size picture ([ShortcutPicture](#) property), the first icon found in the HTML caption is not displayed in the shortcut bar. The entire Shortcut caption is displayed when the shortcut is expanded. Use the [ExpandShortcutCount](#) property to expand the number of shortcuts in the control's shortcut bar.

The Shortcut property supports the following HTML tags:

- ** bold **
- **<u> underline </u>**
- **<s> strikeout </s>**
- **<i> italic </i>**
- **<fgcolor = FF0000> fgcolor </fgcolor>**
- **<bgcolor = FF0000> bgcolor </bgcolor>**
- **
** breaks a line.
- **<solidline>** draws a solid line
- **<dotline>** draws a dotted line
- **<upline>** draws the line to the top of the text line
- **<r>** aligns the rest of the text line to the right side
- **number** inserts an icon inside the item's caption. The number indicates the index of the icon being inserted.
- **key[:width]** inserts a custom size picture being 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.
- **text ** displays portions of text with a different font and/or different size. For instance, the **bit** draws the bit text using the Tahoma font, on size 12 pt. If the name of the font is missing, and instead size is present, the current font is used with a different size. For instance, **bit** displays the bit text using the current font, but with a different size.

Newer HTML format supports subscript and superscript like follows:

- **<off offset> ... </off>** defines the vertical offset to display the text/element. The offset parameter defines the offset to display the element. This tag is inheritable, so the offset is keep while the associated </off> tag is found. You can use the <off offset> HTML tag in combination with the to define a smaller or a larger font to be displayed. For instance: *"Text with <off 6>subscript"* displays the text such as: Text with subscript The *"Text with <off -6>superscript"* displays the text such as: Text with superscript

Also, newer HTML format supports decorative text like follows:

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

gradient-center

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

outlined

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

shadow

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

outline anti-aliasing

property Group.StrikeOut as Boolean

Specifies whether the group's caption should appear in knockout.

Type	Description
Boolean	A boolean expression that specifies whether the group's caption should appear in knockout.

Use the [Bold](#), [Italic](#), [Underline](#) and [StrikeOut](#) properties to apply different font attributes to the group. Use the [Caption](#) property to display different parts of the caption using HTML format. Use the [Font](#) property to specify the control's font. Use the [StrikeOut](#) property to specify whether the item's font is displayed with a horizontal line through it. Use the <s> HTML tag to specify parts of group's caption that should appear in knockout, if [CaptionFormat](#) property is exHTML.

The following VB sample specify that all groups should appear in knockout:

```
Private Sub ListBar1_AddGroup(ByVal Group As EXLISTBARLibCtl.IGroup)
    With Group
        .StrikeOut = True
    End With
End Sub
```

The following C++ sample specify that all groups should appear in knockout:

```
void OnAddGroupListbar1(LPDISPATCH Group)
{
    CGroup group( Group ); group.m_bAutoRelease = FALSE;
    group.SetStrikeOut( TRUE );
}
```

The following VB.NET sample specify that all groups should appear in knockout:

```
Private Sub AxListBar1_AddGroup(ByVal sender As System.Object, ByVal e As
AxEXLISTBARLib._IListBarEvents_AddGroupEvent) Handles AxListBar1.AddGroup
    With e.group
        .StrikeOut = True
    End With
End Sub
```

The following C# sample specify that all groups should appear in knockout:

```
private void axListBar1_AddGroup(object sender,  
AxEXLISTBARLib._IListBarEvents_AddGroupEvent e)  
{  
    e.group.StrikeOut = true;  
}
```

The following VFP sample specify that all groups should appear in strikeouts:

```
*** ActiveX Control Event ***  
LPARAMETERS group  
  
with group  
    .StrikeOut = .t.  
endwith
```

property Group.ToolTip as Variant

Specifies the group's tooltip.

Type	Description
Variant	A string expression that indicates the item's tool tip.

By default, the group's tooltip is empty. If the ToolTip property is empty the control displays no tooltip when the cursor hovers the group's caption. The ToolTip shows up when the cursor hovers the group's caption. Use the [ToolTipDelay](#) to specify the time in ms that passes before the ToolTip appears. In this case no tooltip is displayed when cursor is over the group's caption. Use the [ToolTipWidth](#) property to specify the width of the tooltip window.

The ToolTip supports built-in HTML format that may includes the followings:

- ** bold **
- **<u> underline </u>**
- **<s> strikeouts </s>**
- **<i> italic </i>**
- **<fgcolor = FF0000> fgcolor </fgcolor>**
- **<bgcolor = FF0000> bgcolor </bgcolor>**
- **
** breaks a line.
- **<solidline>** draws a solid line
- **<dotline>** draws a dotted line
- **<upline>** draws the line to the top of the text line
- **<r>** aligns the rest of the text line to the right side
- **number** inserts an icon inside the item's caption. The number indicates the index of the icon being inserted.
- **key[:width]** inserts a custom size picture being 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.
- **text ** displays portions of text with a different font and/or different size. For instance, the **bit** draws the bit text using the Tahoma font, on size 12 pt. If the name of the font is missing, and instead size is present, the current font is used with a different size. For instance, **bit** displays the bit text using the current font, but with a different size.

Newer HTML format supports subscript and superscript like follows:

- **<off offset> ... </off>** defines the vertical offset to display the text/element. The offset parameter defines the offset to display the element. This tag is inheritable, so the

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

Also, newer HTML format supports decorative text like follows:

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

gradient-center

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

outlined

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

shadow

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

outline anti-aliasing

property Group.Underline as Boolean

Specifies whether the group's caption appears in as underlined.

Type	Description
Boolean	A boolean expression that specifies whether the group's caption is underlined.

Use the [Bold](#), [Italic](#), [Underline](#) and [StrikeOut](#) properties to apply different font attributes to the group. Use the [Caption](#) property to display different parts of the caption using HTML format. Use the [Font](#) property to specify the control's font. Use the [Underline](#) property to underline an item.

The following VB sample underlines all groups:

```
Private Sub ListBar1_AddGroup(ByVal Group As EXLISTBARLibCtl.IGroup)
    With Group
        .Underline = True
    End With
End Sub
```

The following C++ sample underlines all groups:

```
void OnAddGroupListbar1(LPDISPATCH Group)
{
    CGroup group( Group ); group.m_bAutoRelease = FALSE;
    group.SetUnderline( TRUE );
}
```

The following VB.NET sample underlines all groups:

```
Private Sub AxListBar1_AddGroup(ByVal sender As System.Object, ByVal e As
AxEXLISTBARLib._IListBarEvents_AddGroupEvent) Handles AxListBar1.AddGroup
    With e.group
        .Underline = True
    End With
End Sub
```

The following C# sample underlines all groups:

```
private void axListBar1_AddGroup(object sender,
```

```
AxEXLISTBARLib._IListBarEvents_AddGroupEvent e)
```

```
{  
    e.group.Underline = true;  
}
```

The following VFP sample underlines all groups:

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

```
LPARAMETERS group
```

```
with group
```

```
    .Underline = .t.
```

```
endwith
```

property Group.UserData as Variant

Specifies an extra data.

Type	Description
Variant	A Variant that indicates the group's extra data

The UserData property associates an extra data to the item. The UserData property is not used by the control. The UserData are of Variant type, so you will be able to save here what ever you want: numbers, objects, strings, and so on. Use the [RemoveGroup](#) event to release any extra data associated to the group, if case.

Groups object

The Groups collection contains a collection of Group objects. Each Group object contains a collection of Item objects. The [Groups](#) property retrieves the control's Groups collection.

Name	Description
Add	Adds a Group object to the collection and returns a reference to the newly created object.
Clear	Removes all objects in the collection.
Count	Returns the number of objects in the collection.
Item	Returns a specific Group from the collection.
ItemByPos	Retrieves the group given its position.
Remove	Removes a specific member from the collection.

method Groups.Add (Caption as String)

Adds a Group object to the collection and returns a reference to the newly created object.

Type	Description
Caption as String	A string expression that indicates the group's caption
Return	Description
Group	A Group object being added to Groups collection.

The Add method adds a new Group object to Groups collection. The [AddGroup](#) event is fired each time when a new group is added to Groups collection. Use the [AddItem](#) method to add new items to the group. The caption may contain built-in HTML tags, if the [CaptionFormat](#) property is exHTML. Use the [Caption](#) property to access the group's caption. Use the [Image](#) property to display a picture in the caption of the group. Use the [BeginUpdate](#) and [EndUpdate](#) methods to maintain performance while adding new groups or new items. Use the [Position](#) property to specify the position of the group.

The following VB sample adds two groups and two items to each group:

```
With ListBar1
    .BeginUpdate
    With .Groups
        With .Add("Group 1")
            .AddItem "Item 1"
            .AddItem "Item 2"
        End With
        With .Add("Group 2")
            .AddItem "Item 1"
            .AddItem "Item 2"
        End With
    End With
    .EndUpdate
End With
```

The following C++ sample adds two groups and two items to each group:

```
#include "Item.h"
#include "Group.h"
#include "Groups.h"
COleVariant vtMissing; V_VT( &vtMissing ) = VT_ERROR;
```

```
m_listbar.BeginUpdate();
CGroups groups = m_listbar.GetGroups();
CGroup group1 = groups.Add( "Group 1" );
group1.AddItem( "Item 1", vtMissing );
group1.AddItem( "Item 2", vtMissing );
CGroup group2 = groups.Add( "Group 2" );
group2.AddItem( "Item 1", vtMissing );
group2.AddItem( "Item 2", vtMissing );
m_listbar.EndUpdate();
```

The following VB.NET sample adds two groups and two items to each group:

```
With AxListBar1
    .BeginUpdate()
    With .Groups
        With .Add("Group 1")
            .AddItem("Item 1")
            .AddItem("Item 2")
        End With
        With .Add("Group 2")
            .AddItem("Item 1")
            .AddItem("Item 2")
        End With
    End With
    .EndUpdate()
End With
```

The following C# sample adds two groups and two items to each group:

```
axListBar1.BeginUpdate();
EXLISTBARLib.Group group1 = axListBar1.Groups.Add("Group 1");
group1.AddItem("Item 1", null);
group1.AddItem("Item 2", null);
EXLISTBARLib.Group group2 = axListBar1.Groups.Add("Group 2");
group2.AddItem("Item 1", null);
group2.AddItem("Item 2", null);
axListBar1.EndUpdate();
```

The following VFP sample adds two groups and two items to each group:

```
With thisform.ListBar1
  .BeginUpdate()
  With .Groups
    With .Add("Group 1")
      .AddItem("Item 1")
      .AddItem("Item 2")
    EndWith
    With .Add("Group 2")
      .AddItem("Item 1")
      .AddItem("Item 2")
    EndWith
  EndWith
  .EndUpdate()
EndWith
```


method Groups.Clear ()

Removes all objects in the collection.

Type	Description
------	-------------

Use the Clear method to clear the Groups collection. Use the [Remove](#) method to remove a specific group. The [RemoveGroup](#) event is fired when user removes a group. Use the RemoveGroup event to release any extra data that you have associated to a group. When removing a group, all items inside are removed too. Use the [RemoveItem](#) method to remove an item. Use the [GroupHeight](#) property to specify the height of the group's caption.

property Groups.Count as Long

Returns the number of objects in the collection.

Type	Description
Long	A long expression that specifies the count of Group objects into Groups collection.

Counts the [Group](#) objects in the Groups collection. Use the [Item](#) property to access the group giving its index or caption. Use the [Caption](#) property to get the group's caption. Use the [Index](#) property to specify the index of the group. Use the [ItemByPos](#) property to retrieve a group by its position. Use the [Position](#) property to specify the group's position.

The following VB sample enumerates the groups in the control:

```
With ListBar1.Groups
  Dim i As Long
  For i = 0 To .Count - 1
    With .Item(i)
      Debug.Print (.Caption)
    End With
  Next
End With
```

The following VB sample enumerates the groups in the control:

```
Dim g As EXLISTBARLibCtl.Group
For Each g In ListBar1.Groups
  Debug.Print g.Caption
Next
```

The following C++ sample enumerates the groups in the control:

```
CGroups groups = m_listbar.GetGroups();
for ( long i = 0; i < groups.GetCount(); i++ )
{
  CGroup group( groups.GetItem( COleVariant( long(i) ) ) );
  OutputDebugString( group.GetCaption() );
}
```

The following VB.NET sample enumerates the groups in the control:

```
With AxListBar1.Groups
    Dim i As Integer
    For i = 0 To .Count - 1
        With .Item(i)
            Debug.WriteLine(.Caption)
        End With
    Next
End With
```

The following VB.NET sample enumerates the groups in the control:

```
Dim g As EXLISTBARLib.Group
For Each g In AxListBar1.Groups
    Debug.WriteLine(g.Caption)
Next
```

The following C# sample enumerates the groups in the control:

```
for (int i = 0; i < axListBar1.Groups.Count; i++)
{
    EXLISTBARLib.Group g = axListBar1.Groups[i];
    System.Diagnostics.Debug.WriteLine(g.Caption);
}
```

The following VFP sample enumerates the groups in the control:

```
With thisform.ListBar1.Groups
    local i
    For i = 0 To .Count - 1
        With .Item(i)
            wait window nowait .Caption
        EndWith
    Next
EndWith
```

property Groups.Item (Index as Variant) as Group

Returns a specific Group from the collection.

Type	Description
Index as Variant	A long expression that indicates the group's index, or a string expression that indicates the group's caption.
Group	A Group object being retrieved.

Use the Item property to access a given Group object. The Item property is the default property in the Groups object, and Groups.Item(x) is similar with Groups(x). Use the [Count](#) property to count the groups in the control. Use the [Caption](#) property to get the group's caption. Use the [ItemByPos](#) property to access a Group object by its position.

The following VB sample enumerates the groups in the control:

```
With ListBar1.Groups
    Dim i As Long
    For i = 0 To .Count - 1
        With .Item(i)
            Debug.Print (.Caption)
        End With
    Next
End With
```

The following VB sample enumerates the groups in the control:

```
Dim g As EXLISTBARLibCtl.Group
For Each g In ListBar1.Groups
    Debug.Print g.Caption
Next
```

The following C++ sample enumerates the groups in the control:

```
CGroups groups = m_listbar.GetGroups();
for ( long i = 0; i < groups.GetCount(); i++ )
{
    CGroup group( groups.GetItem( COleVariant( long(i) ) ) );
    OutputDebugString( group.GetCaption() );
}
```

The following VB.NET sample enumerates the groups in the control:

```
With AxListBar1.Groups
    Dim i As Integer
    For i = 0 To .Count - 1
        With .Item(i)
            Debug.WriteLine(.Caption)
        End With
    Next
End With
```

The following VB.NET sample enumerates the groups in the control:

```
Dim g As EXLISTBARLib.Group
For Each g In AxListBar1.Groups
    Debug.WriteLine(g.Caption)
Next
```

The following C# sample enumerates the groups in the control:

```
for (int i = 0; i < axListBar1.Groups.Count; i++)
{
    EXLISTBARLib.Group g = axListBar1.Groups[i];
    System.Diagnostics.Debug.WriteLine(g.Caption);
}
```

The following VFP sample enumerates the groups in the control:

```
With thisform.ListBar1.Groups
    local i
    For i = 0 To .Count - 1
        With .Item(i)
            wait window nowait .Caption
        EndWith
    Next
EndWith
```

property Groups.ItemByPos (Position as Long) as Group

Retrieves the group given its position.

Type	Description
Position as Long	A long expression that indicates the position of the requested group
Group	A Group object being retrieved.

Use the ItemByPos property to access the Group object by its position. Use the [Position](#) property to specify the group's position. Use the [Caption](#) property to get the group's caption. Use the [Item](#) property to access a given Group object. Use the [Count](#) property to count the groups in the control.

The following VB sample enumerates the groups in the control, as they are displayed:

```
With ListBar1.Groups
  Dim i As Long
  For i = 0 To .Count - 1
    With .ItemByPos(i)
      Debug.Print (.Caption)
    End With
  Next
End With
```

The following C++ sample enumerates the groups in the control, as they are displayed:

```
CGroups groups = m_listbar.GetGroups();
for ( long i = 0; i < groups.GetCount(); i++ )
{
  CGroup group( groups.GetItemByPos( i ) );
  OutputDebugString( group.GetCaption() );
}
```

The following VB.NET sample enumerates the groups in the control, as they are displayed:

```
With AxListBar1.Groups
  Dim i As Integer
  For i = 0 To .Count - 1
    With .ItemByPos(i)
```

```
        Debug.WriteLine(.Caption)
    End With
Next
End With
```

The following C# sample enumerates the groups in the control, as they are displayed:

```
for (int i = 0; i < axListBar1.Groups.Count; i++)
{
    EXLISTBARLib.Group g = axListBar1.Groups.get_ItemByPos(i);
    System.Diagnostics.Debug.WriteLine(g.Caption);
}
```

The following VFP sample enumerates the groups in the control, as they are displayed:

```
With thisform.ListBar1.Groups
    local i
    For i = 0 To .Count - 1
        With .ItemByPos(i)
            wait window nowait .Caption
        EndWith
    Next
EndWith
```

method Groups.Remove (Index as Variant)

Removes a specific member from the collection.

Type	Description
Index as Variant	A long expression that indicates the the group's index, or a string expression that indicates the group's caption.

Use the Remove method to remove a specific Group object. The [RemoveGroup](#) event is fired when the user removes a group. Use the [Clear](#) method to clear the entire Groups collection. The items in a group are removed too, when the Groups is removed. Use the [RemoveItem](#) method to remove an item from the group. Use the [Index](#) property to retrieve the index of the group. Use the [Caption](#) property to specify the caption of the group.

Item object

The Item object holds information about the control's item. The Item object supports the following properties:

Name	Description
Alignment	Specifies the item's alignment.
BackColor	Retrieves or sets the item's background color.
BackColor2	Specifies the color at the ending boundary line of the gradient item's caption.
Bold	Specifies whether the item's caption should appear in bold.
Caption	Specifies the item's caption.
CaptionFormat	Specifies how the item's caption is displayed.
ForeColor	Specifies the item's foreground color.
Group	Gets the owner group.
Image	Specifies the item's image.
Indent	Specifies the amount in pixels of the item's indent.
Index	Gets the index of the item.
Italic	Specifies whether the item's caption should appear in italic.
Position	Specifies the item's position.
StrikeOut	Specifies whether the item's caption should appear in strikeout.
ToolTip	Specifies the item's tooltip.
Underline	Specifies whether the item's caption appears as underlined..
UserData	Associates an extra data to the object.
Visible	Specifies whether the item is visible or hidden.

property Item.Alignment as AlignmentEnum

Specifies the item's alignment.

Type	Description
AlignmentEnum	An AlignmentEnum expression that indicates the item's alignment.

By default, the item's alignment is exCenter. Use the [AddItem](#) event to change the alignment for all items into a group, like in the following samples. Use the [ItemHeight](#) property to specify the height for all items in the control. Use the [Image](#) property to assign a picture to an item. Use the [Caption](#) property to specify the caption of the item.

The following VB sample changes the item's alignment when a new items is added to the first group:

```
Private Sub ListBar1_AddItem(ByVal Item As EXLISTBARLibCtl.IItem)
    With Item
        If (.Group.Index = 0) Then
            .Alignment = exRight
        End If
    End With
End Sub
```

The following C++ sample changes the item's alignment when a new items is added to the first group:

```
void OnAddItemListbar1(LPDISPATCH Item)
{
    CItem item( Item ); item.m_bAutoRelease = FALSE;
    if ( item.GetGroup().GetIndex() == 0 )
        item.SetAlignment( 2 /*exRight*/ );
}
```

The following VB.NET sample changes the item's alignment when a new items is added to the first group:

```
Private Sub AxListBar1_AddItem(ByVal sender As Object, ByVal e As
AxEXLISTBARLib._IListBarEvents_AddItemEvent) Handles AxListBar1.AddItem
    With e.item
        If (.Group.Index = 0) Then
```

```
.Alignment = EXLISTBARLib.AlignmentEnum.exRight
End If
End With
End Sub
```

The following C# sample changes the item's alignment when a new items is added to the first group:

```
private void axListBar1_AddItem(object sender,
AxEXLISTBARLib._IListBarEvents_AddItemEvent e)
{
    if (e.item.Group.Index == 0)
        e.item.Alignment = EXLISTBARLib.AlignmentEnum.exRight;
}
```

The following VFP sample changes the item's alignment when a new items is added to the first group:

```
*** ActiveX Control Event ***
LPARAMETERS item

with item
    If (.Group.Index = 0) Then
        .Alignment = 2 && exRight
    EndIf
endwith
```

property Item.BackColor as Color

Retrieves or sets the item's background color.

Type	Description
Color	A color expression that indicates the item'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.

Use the BackColor and [ForeColor](#) properties to specify the item's background and foreground colors. The [BackColor2](#) property specifies the color at the ending boundary line of the gradient item's caption. Use the [BackColorList](#) property to specify the default background color for the group's list. Use the <bgcolor> built in HTML tag in the [Caption](#) property to define portions of text using a specified background color.

In VB.NET or C# you require the following functions until the .NET framework will provide:

You can use the following VB.NET function:

```
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
```

You can use the following C# function:

```
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 VB sample changes the item's background color:

```
With ListBar1.Groups(0).Item(0)  
    .BackColor = vbBlue  
End With
```

The following C++ sample changes the item's background color:

```
m_listbar.GetGroups().GetItem( COleVariant( long(0) ) ).GetItem( COleVariant( long(0) )  
)>.SetBackColor( RGB(0,0,255) );
```

The following VB.NET sample changes the item's background color:

```
With AxListBar1.Groups(0).Item(0)  
    .BackColor = ToUInt32(Color.Blue)  
End With
```

The following C# sample changes the item's background color:

```
axListBar1.Groups[0][0].BackColor = ToUInt32(Color.Blue);
```

The following VFP sample changes the item's background color:

```
With thisform.ListBar1.Groups(0).Item(0)  
    .BackColor = RGB(0,0,255)  
EndWith
```

property Item.BackColor2 as Color

Specifies the color at the ending boundary line of the gradient item's caption.

Type	Description
Color	A color expression that indicates the color at the ending boundary line of the gradient item's caption.

Use the BackColor2 property to specify the second background color when painting its background in gradient. Use the [BackColor](#) and [ForeColor](#) properties to specify the item's background and foreground colors. Use the [BackColorList](#) property to specify the default background color for the group's list. Use the [BackColor](#) property to specify the control's background color.

In VB.NET or C# you require the following functions until the .NET framework will provide:

You can use the following VB.NET function:

```
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
```

You can use the following C# function:

```
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);
}
```

property Item.Bold as Boolean

Specifies whether the item's caption should appear in bold.

Type	Description
Boolean	A boolean expression that indicates whether the item's caption should appear in bold.

Use the Bold, [Italic](#), [Underline](#) and [StrikeOut](#) properties to apply different font attributes to the item. Use the [Caption](#) property to display different parts of the caption using HTML format. Use the [Font](#) property to specify the control's font. Use the [Group](#) property to get the group that owns the item.

The following VB sample bolds all items in the first group:

```
Private Sub ListBar1_AddItem(ByVal Item As EXLISTBARLibCtl.IItem)
    If (Item.Group.Index = 0) Then
        With Item
            .Bold = True
        End With
    End If
End Sub
```

The following C++ sample bolds all items in the first group:

```
void OnAddItemListbar1(LPDISPATCH Item)
{
    CItem item( Item ); item.m_bAutoRelease = FALSE;
    if ( item.GetGroup().GetIndex() == 0 )
        item.SetBold( TRUE );
}
```

The following VB.NET sample bolds all items in the first group:

```
Private Sub AxListBar1_AddItem(ByVal sender As Object, ByVal e As
AxEXLISTBARLib._IListBarEvents_AddItemEvent) Handles AxListBar1.AddItem
    With e.item
        If (.Group.Index = 0) Then
            .Bold = True
        End If
    End With
End Sub
```

```
End With  
End Sub
```

The following C# sample bolds all items in the first group:

```
private void axListBar1_AddItem(object sender,  
AxEXLISTBARLib._IListBarEvents_AddItemEvent e)  
{  
    if (e.item.Group.Index == 0)  
        e.item.Bold = true;  
}
```

The following VFP sample bolds all items in the first group:

```
*** ActiveX Control Event ***  
LPARAMETERS item  
  
with item  
    If (.Group.Index = 0) Then  
        .Bold = .t.  
    EndIf  
endwith
```


property Item.Caption as String

Specifies the item's caption.

Type	Description
String	A string expression that indicates the item's caption.

Use the Caption property to change the item's caption. Use the [UserData](#) property to associate an extra data to the item. Use the [CaptionFormat](#) property to allow built-in HTML tags in the cell's caption. Use the [Bold](#), [Italic](#), [Underline](#) or [StrikeOut](#) property to define the font for caption of the item. The control fires the [SelectItem](#) event when the user clicks the item. You can specify the item's caption when using the [AddItem](#) method. Use the [ItemHeight](#) property to specify the height of the items in the group. Use the **** HTML tag to insert icons inside the item's caption, if the [CaptionFormat](#) property is exHTML.

property Item.CaptionFormat as CaptionFormatEnum

Specifies how the item's caption is displayed.

Type	Description
CaptionFormatEnum	A CaptionFormatEnum expression that indicates whether the control uses built-in HTML tags to display the cell's caption.

Use the CaptionFormat property to allow built-in HTML tags in the item's caption. Use the [Caption](#) property to specifies the cell's caption. By default, the CaptionFormat property is exText.

The control supports the following HTML tags:

- ** bold **
- **<u> underline </u>**
- **<s> strikeout </s>**
- **<i> italic </i>**
- **<fgcolor = FF0000> fgcolor </fgcolor>**
- **<bgcolor = FF0000> bgcolor </bgcolor>**
- **
** breaks a line.
- **<solidline>** draws a solid line
- **<dotline>** draws a dotted line
- **<upline>** draws the line to the top of the text line
- **<r>** aligns the rest of the text line to the right side
- **number** inserts an icon inside the item's caption. The number indicates the index of the icon being inserted.
- **key[:width]** inserts a custom size picture being 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.
- **text ** displays portions of text with a different font and/or different size. For instance, the **bit** draws the bit text using the Tahoma font, on size 12 pt. If the name of the font is missing, and instead size is present, the current font is used with a different size. For instance, **bit** displays the bit text using the current font, but with a different size.

Also, newer HTML format supports decorative text like follows:

- **<gra rrrgggbb;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 FFFFFFFF;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 Item.ForeColor as Color

Specifies the item's foreground color.

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

Use the [BackColor](#) and ForeColor properties to specify the item's background and foreground colors. Use the [ForeColorList](#) property to specify the default foreground color for the group's list. Use the [ForeColor](#) property to specify the control's foreground color. Use the <fgcolor> built in HTML tag in the [Caption](#) property to define portions of text using a specified foreground color.

In VB.NET or C# you require the following functions until the .NET framework will provide:

You can use the following VB.NET function:

```
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
```

You can use the following C# function:

```
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 VB sample changes the item's foreground color:

```
With ListBar1.Groups(0).Item(0)
    .ForeColor = vbBlue
```

End With

The following C++ sample changes the item's foreground color:

```
m_listbar.GetGroups().GetItem( COleVariant( long(0) ) ).GetItem( COleVariant( long(0) ) ).SetForeColor( RGB(0,0,255) );
```

The following VB.NET sample changes the item's foreground color:

```
With AxListBar1.Groups(0).Item(0)  
    .ForeColor = ToUInt32(Color.Blue)  
End With
```

The following C# sample changes the item's foreground color:

```
axListBar1.Groups[0][0].ForeColor = ToUInt32(Color.Blue);
```

The following VFP sample changes the item's foreground color:

```
With thisform.ListBar1.Groups(0).Item(0)  
    .ForeColor = RGB(0,0,255)  
EndWith
```

property Item.Group as Group

Gets the owner group.

Type	Description
Group	A Group object that's the owner of the item.

The Group property specify the owner group for the item. Use the [AddItem](#) method to add new items to a group. Use the [Item](#) property to retrieve an item from a group giving its index or its caption. Use the [ItemByPos](#) property to retrieve an item from a group giving its position. Use the [Position](#) property to specify the item's position. Use the [RemoveItem](#) property to remove an item from a group.

property Item.Image as Variant

Specifies the item's icon.

Type	Description
Variant	A long expression that indicates the index of icon being used, a string expression that indicates the base64 encoded string that holds a picture object, or a Picture object. Use the eximages tool to save your picture as base64 encoded format.

By default, the Image property is 0. The images list collection is 1 based. Use the Image property to assign an icon or a picture to an item. Use the Image parameter of the [AddItem](#) method to assign an image at adding time. Use the [Replacelcon](#) and [Images](#) method to change the icons list collection at runtime. Use the **** HTML tag to insert icons inside the item's caption, if the [CaptionFormat](#) property is exHTML. At design time, the control displays an icons list window (if the [ShowImageList](#) property is true) where icons or resource files can be dropped.

The following VB sample loads a collection of icons from a BASE64 encoded string:

```
With ListBar1
  Dim s As String
  s =
" gBJJgBAIEAAGAEGCAAhb/hz/EIAh8Tf5CJo2AEZjQAjEZFEaIEaEEaAIAkcbk0oIUrkltl0vmExml

  s = s +
"FphZDEJtT1zp7bd1XasiLB8ld4o8kCRJIACSpRfNKWXd1uH+eaHn8P55necB/kmf5+A+eBnr

  .BeginUpdate
  .Images s
  With .Groups
    With .Add("Group 1")
      .AddItem "Item 1", 1
      .AddItem "Item 2", 2
    End With
    With .Add("Group 2")
      .AddItem "Item 1", 2
      .AddItem "Item 2", 1
    End With
  End With
End With
```

End With
.EndUpdate
End With

The following C++ sample loads a collection of icons from a BASE64 encoded string:

```
#include "Item.h"
#include "Group.h"
#include "Groups.h"
COleVariant vtMissing; V_VT( &vtMissing ) = VT_ERROR;
CString
s("gBJJgBAIEAAGAEGCAAhb/hz/EIAh8Tf5CJo2AEZjQAjEZFEalEaEEaAlAkcbk0oIUrlktl0vmExr

s = s +
"FphZDEJtT1zp7bd1XasiLB8ld4o8kCRJIACSpRfNKWXd1uH+eaHn8P55necB/kmf5+A+eBnr

m_listbar.BeginUpdate();
CGroups groups = m_listbar.GetGroups();
m_listbar.Images(COleVariant(s));
CGroup group1 = groups.Add( "Group 1" );
group1.AddItem( "Item 1", COleVariant(long(1)) );
group1.AddItem( "Item 2", COleVariant(long(2)) );
CGroup group2 = groups.Add( "Group 2" );
group2.AddItem( "Item 1", COleVariant(long(3)) );
group2.AddItem( "Item 2", COleVariant(long(4)) );
m_listbar.EndUpdate();
```

The following VB.NET sample loads a collection of icons from a BASE64 encoded string:

```
With AxListBar1
    Dim s As String =
    "gBJJgBAIEAAGAEGCAAhb/hz/EIAh8Tf5CJo2AEZjQAjEZFEalEaEEaAlAkcbk0oIUrlktl0vmExml

    s = s +
    "FphZDEJtT1zp7bd1XasiLB8ld4o8kCRJIACSpRfNKWXd1uH+eaHn8P55necB/kmf5+A+eBnr

    .BeginUpdate()
    .Images(s)
```



```
With .Groups
```

```
    With .Add("Group 1")
```

```
        .AddItem("Item 1", 1)
```

```
        .AddItem("Item 2", 2)
```

```
    End With
```

```
    With .Add("Group 2")
```

```
        .AddItem("Item 1", 3)
```

```
        .AddItem("Item 2", 4)
```

```
    End With
```

```
End With
```

```
.EndUpdate()
```

```
End With
```

The following C# sample loads a collection of icons from a BASE64 encoded string:

```
String s=
```

```
"gBJJgBAIEAAGAEGCAAhb/hz/EIAh8Tf5CJo2AEZjQAjEZFEaIEaEEaAIAkcbk0oIUrlktl0vmExml
```

```
s = s +
```

```
"FphZDEJtT1zp7bd1XasiLB8ld4o8kCRJIACSpRfNKWXd1uH+eaHn8P55necB/kmf5+A+eBnr
```

```
axListBar1.BeginUpdate();
```

```
axListBar1.Images(s);
```

```
EXLISTBARLib.Group group1 = axListBar1.Groups.Add("Group 1");
```

```
group1.AddItem("Item 1", 1);
```

```
group1.AddItem("Item 2", 2);
```

```
EXLISTBARLib.Group group2 = axListBar1.Groups.Add("Group 2");
```

```
group2.AddItem("Item 1", 3);
```

```
group2.AddItem("Item 2", 4);
```

```
axListBar1.EndUpdate();
```

The following VFP sample loads a collection of icons from a BASE64 encoded string:

```
With thisform.ListBar1
```

```
    local s
```

```
    .BeginUpdate()
```

```
    s =
```

```
"gBJJgBAIEAAJAEGCAAhb/hz/EIAh8Tf5CJo2AEZjQAjEZFEaIEaGEaAIAEEbjMjlErIktl0vmExmU
```

```

s = s +
"Fw2HxGJxWLxmNx2PyGRyWTymVy2XzGZzWbzmdz2f0Gh0Wj0ml02npqAQEZ1WojWq1b.

s = s +
"Gjb8wQ4L1us7zeQUmbqP29z4uu3rWv+ +jVwo7kKus3MNRA1cOQk6SgRPEkMwu3EXRY9'

s = s +
"A/koxFKE6Q3O6WT9M8MSHQ0MuXQLpSXLE3xZJUwJy8bW0W+UPT7DEnUIMEa0xHdMC

s = s +
"pdLzXZfsNX8+qfUEs2Bxg5lJwerN7Jld7LWlz2Atc0qLB8lmKo8kCRJIACSo2QGO11hOJM2hx

s = s + "+YJ/7YPaHnGiaAg="
With .Groups
    With .Add("Group 1")
        .AddItem("Item 1",1)
        .AddItem("Item 2",2)
    EndWith
    With .Add("Group 2")
        .AddItem("Item 1",3)
        .AddItem("Item 2",4)
    EndWith
EndWith
.EndUpdate()
EndWith

```

The following Template sample loads a collection of icons from a BASE64 encoded string:

```

BeginUpdate
Images("gBJJgBAIEAAJAEGCAAhb/hz/EIAh8Tf5CJo2AEZjQAjEZFEaIEaGEaAIAEEbjMjIErIktlOv

Groups
{
    "Group 1"
    {
        AddItem("Item 1",1)
    }
}

```

```
        AddItem("Item 2",2)
    }
    "Group 2"
    {
        AddItem("Item 1",3)
        AddItem("Item 2",4)
    }
}
EndUpdate
```

property Item.Indent as Long

Specifies the amount in pixels of the item's indent.

Type	Description
Long	A long expression that specifies the amount in pixels of the item's indent.

By default, the item's indent is zero. Use the Indent property to indent items, so they look like a tree. Use the [Caption](#) property to specify the caption of the item. Use the [Image](#) property to assign a picture to an item. The Indent property indents the caption and the image too. For instance, if the item is right aligned, the indentation is on the right side of the group.

property Item.Index as Long

Gets the index of the item.

Type	Description
Long	A long expression that indicates the item's index.

The Index property gets the index of the Item object in items collection of the owner group. Use the [Position](#) property to change the item's position. The Index property is allocated when the items is added to the group's list, using the [AddItem](#) method. Use the [Item](#) property to access an item giving its index of by its caption. Use the [Caption](#) property to specify the caption of the item. Use the [Image](#) property to assign a picture to an item.

property Item.Italic as Boolean

Specifies whether the item's caption should appear in italic.

Type	Description
Boolean	A boolean expression that indicates whether the item's caption should appear in italic.

Use the [Bold](#), [Italic](#), [Underline](#) and [StrikeOut](#) properties to apply different font attributes to the item. Use the [Caption](#) property to display different parts of the caption using HTML format. Use the [Font](#) property to specify the control's font. Use the [Group](#) property to get the group that owns the item.

The following VB sample specifies that all items in the first group should appear in italic:

```
Private Sub ListBar1_AddItem(ByVal Item As EXLISTBARLibCtl.IItem)
    If (Item.Group.Index = 0) Then
        With Item
            .Italic = True
        End With
    End If
End Sub
```

The following C++ sample specifies that all items in the first group should appear in italic:

```
void OnAddItemListbar1(LPDISPATCH Item)
{
    CItem item( Item ); item.m_bAutoRelease = FALSE;
    if ( item.GetGroup().GetIndex() == 0 )
        item.SetItalic( TRUE );
}
```

The following VB.NET sample specifies that all items in the first group should appear in italic:

```
Private Sub AxListBar1_AddItem(ByVal sender As Object, ByVal e As
AxEXLISTBARLib._IListBarEvents_AddItemEvent) Handles AxListBar1.AddItem
    With e.item
        If (.Group.Index = 0) Then
            .Italic = True
        End If
    End With
End Sub
```

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

The following C# sample specifies that all items in the first group should appear in italic:

```
private void axListBar1_AddItem(object sender,
AxEXLISTBARLib._IListBarEvents_AddItemEvent e)
{
    if (e.item.Group.Index == 0)
        e.item.Italic = true;
}
```

The following VFP sample specifies that all items in the first group should appear in italic:

```
*** ActiveX Control Event ***
LPARAMETERS item

with item
    If (.Group.Index = 0) Then
        .Italic = .t.
    EndIf
endwith
```

property Item.Position as Long

Specifies the item's position.

Type	Description
Long	A long expression that indicates the item's position.

Use the Position property to change the item's position. Use the [Index](#) property to retrieve the item's index. Use the [ItemByPos](#) property to retrieve an item giving its position. Use the [ItemHeight](#) property to specify the height for all items in the group. Use the [Visible](#) property to hide an item. The control adds the item to the end of the list, when the [AddItem](#) method is called.

property Item.StrikeOut as Boolean

Specifies whether the item's caption should appear in strikeout.

Type	Description
Boolean	A boolean expression that indicates whether the item's caption should appear in strikeout

Use the Bold, [Italic](#), [Underline](#) and [StrikeOut](#) properties to apply different font attributes to the item. Use the [Caption](#) property to display different parts of the caption using HTML format. Use the [Font](#) property to specify the control's font. Use the [Group](#) property to get the group that owns the item.

The following VB sample specifies each item in the first group displays a horizontal line through it:

```
Private Sub ListBar1_AddItem(ByVal Item As EXLISTBARLibCtl.IItem)
    If (Item.Group.Index = 0) Then
        With Item
            .StrikeOut = True
        End With
    End If
End Sub
```

The following C++ sample specifies each item in the first group displays a horizontal line through it:

```
void OnAddItemListbar1(LPDISPATCH Item)
{
    CItem item( Item ); item.m_bAutoRelease = FALSE;
    if ( item.GetGroup().GetIndex() == 0 )
        item.SetStrikeOut( TRUE );
}
```

The following VB.NET sample specifies each item in the first group displays a horizontal line through it:

```
Private Sub AxListBar1_AddItem(ByVal sender As Object, ByVal e As
AxEXLISTBARLib._IListBarEvents_AddItemEvent) Handles AxListBar1.AddItem
    With e.item
        If (.Group.Index = 0) Then
```

```
.StrikeOut = True
End If
End With
End Sub
```

The following C# sample specifies each item in the first group displays a horizontal line through it:

```
private void axListBar1_AddItem(object sender,
AxEXLISTBARLib._IListBarEvents_AddItemEvent e)
{
    if (e.item.Group.Index == 0)
        e.item.StrikeOut = true;
}
```

The following VFP sample specifies each item in the first group displays a horizontal line through it:

```
*** ActiveX Control Event ***
LPARAMETERS item

with item
    If (.Group.Index = 0) Then
        .StrikeOut = .t.
    EndIf
endwith
```

property Item.ToolTip as Variant

Specifies the item's tooltip.

Type	Description
Variant	A string expression that indicates the item's tool tip.

The item's tooltip is displayed whenever the cursor is over the item. The [ToolTipPopDelay](#) property specifies the time in ms that passes before the ToolTip appears. Use the [ToolTip](#) property to specify the tooltip for the group. Use the [ToolTipDelay](#) property to specify the time in ms that passes before the ToolTip appears. In this case the control displays no tooltip when the cursor is over the item. Use the [ToolTipWidth](#) property to specify the width of the tooltip window.

The ToolTip supports built-in HTML format that may includes the followings:

- ** bold **
- **<u> underline </u>**
- **<s> strikeouts </s>**
- **<i> italic </i>**
- **<fgcolor = FF0000> fgcolor </fgcolor>**
- **<bgcolor = FF0000> bgcolor </bgcolor>**
- **
** breaks a line.
- **<solidline>** draws a solid line
- **<dotline>** draws a dotted line
- **<upline>** draws the line to the top of the text line
- **<r>** aligns the rest of the text line to the right side
- **number** inserts an icon inside the item's caption. The number indicates the index of the icon being inserted.
- **key[:width]** inserts a custom size picture being 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.
- **text ** displays portions of text with a different font and/or different size. For instance, the **bit** draws the bit text using the Tahoma font, on size 12 pt. If the name of the font is missing, and instead size is present, the current font is used with a different size. For instance, **bit** displays the bit text using the current font, but with a different size.

Also, newer HTML format supports decorative text like follows:

- **<gra rrrgggbb;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 FFFFFFFF;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 Item.Underline as Boolean

Specifies whether the item's caption should appear as underlined.

Type	Description
Boolean	A color expression that indicates whether the item's caption is underlined.

Use the Bold, [Italic](#), Underline and [StrikeOut](#) properties to apply different font attributes to the item. Use the [Caption](#) property to display different parts of the caption using HTML format. Use the [Font](#) property to specify the control's font. Use the [Group](#) property to get the group that owns the item.

The following VB sample underlines all items in the first group:

```
Private Sub ListBar1_AddItem(ByVal Item As EXLISTBARLibCtl.IItem)
    If (Item.Group.Index = 0) Then
        With Item
            .Underline = True
        End With
    End If
End Sub
```

The following C++ sample underlines all items in the first group:

```
void OnAddItemListbar1(LPDISPATCH Item)
{
    CItem item( Item ); item.m_bAutoRelease = FALSE;
    if ( item.GetGroup().GetIndex() == 0 )
        item.SetUnderline( TRUE );
}
```

The following VB.NET sample underlines all items in the first group:

```
Private Sub AxListBar1_AddItem(ByVal sender As Object, ByVal e As
AxEXLISTBARLib._IListBarEvents_AddItemEvent) Handles AxListBar1.AddItem
    With e.item
        If (.Group.Index = 0) Then
            .Underline = True
        End If
    End With
End Sub
```

```
End With
End Sub
```

The following C# sample underlines all items in the first group:

```
private void axListBar1_AddItem(object sender,
AxEXLISTBARLib._IListBarEvents_AddItemEvent e)
{
    if (e.item.Group.Index == 0)
        e.item.Underline = true;
}
```

The following VFP sample underlines all items in the first group:

```
*** ActiveX Control Event ***
LPARAMETERS item

with item
    If (.Group.Index = 0) Then
        .Underline = .t.
    EndIf
endwith
```

property Item.userData as Variant

Associates an extra data to the object.

Type	Description
Variant	A Variant that specifies the item's user data.

The UserData property associates an extra data to the item. The UserData property is not used by the control. The UserData are of Variant type, so you will be able to save here what ever you want: numbers, objects, strings, and so on. Use the [RemoveItem](#) event to release any extra data associated to an item, if case.

property Item.Visible as Boolean

Specifies whether the item is visible or hidden.

Type	Description
Boolean	A boolean expression that indicates whether the item is visible or hidden.

Use the Visible property to show or hide an item. Use the [Position](#) property to specify the item's position. Use the [RemoveItem](#) method to remove an item from the group. Use the [Caption](#) property to specify the caption of the item. Use the [ItemByPos](#) property to retrieve an item giving its position. Use the Image property to add or remove an icon or a picture to the item. Use the [ItemHeight](#) property to specify the height for all items in the group.

The following VB sample hides the first item in the first group:

```
With ListBar1.Groups(0).ItemByPos(0)
    .Visible = False
End With
```

The following C++ sample hides the first item in the first group:

```
#include "Item.h"
#include "Group.h"
#include "Groups.h"
CGroups groups = m_listbar.GetGroups();
CGroup group = groups.GetItem( COleVariant(long(0)) );
CItem item = group.GetItemByPos( 0 );
item.SetVisible( FALSE );
```

The following VB.NET sample hides the first item in the first group:

```
With AxListBar1.Groups(0).ItemByPos(0)
    .Visible = False
End With
```

The following C# sample hides the first item in the first group:

```
axListBar1.Groups[0].get_ItemByPos(0).Visible = false;
```

The following VFP sample hides the first item in the first group:


```
with thisform.ListBar1
  with .Groups.Item(0)
    with .ItemByPos(0)
      .Visible = .f
    endwith
  endwith
endwith
```

ListBar 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: {41387A8B-6293-46CE-B9D8-55F49AE0DA60}. The object's program identifier is: "Exontrol.ListBar". The /COM object module is: "ExListBar.dll"

The Exontrol's ExListBar ActiveX control, an accurate reproduction of the Microsoft Outlook Bar, provides an intuitive user-interface when large amounts of information need to be presented. 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 ExListBar supports the following properties and methods:

Name	Description
AllowResizeShortcutBar	Specifies whether the user can resize the shortcutbar, to allow multiple shortcuts to be visible.
AnchorFromPoint	Retrieves the identifier of the anchor from point.
Appearance	Specifies the control's appearance.
AttachTemplate	Attaches a script to the current object, including the events, from a string, file, a safe array of bytes.
BackColor	Retrieves or sets a value that indicates the control's background color.
BackColorGroup	Retrieves or sets a value that indicates the group's background color.
BackColorGroup2	Specifies the color at the ending boundary line of the gradient group's caption.
Background	Returns or sets a value that indicates the background color for parts in the control.
BeginUpdate	Maintains performance when items are added to the control one at a time.
BorderHeight	Specifies the border's height.
BorderWidth	Specifies the border's width.
DelayScroll	Specifies the delay used when user selects a group.
EndUpdate	Resumes painting the control after painting is suspended by the BeginUpdate method.
EventParam	Retrieves or sets a value that indicates the current's event parameter.
ExecuteTemplate	Executes a template and returns the result.

ExpandShortcutCount	Retrieves or sets a value that indicates the number of shortcuts being expanded.
ExpandShortcutImage	Retrieves or sets a value that indicates the index of the image being displayed to expand the shortcuts.
Font	Retrieves or sets the control's font.
ForeColor	Retrieves or sets a value that indicates the control's foreground color.
ForeColorGroup	Retrieves or sets a value that indicates the group's foreground color.
FormatAnchor	Specifies the visual effect for anchor elements in HTML captions.
GroupAppearance	Specifies the group's appearance.
GroupFromPoint	Gets the group from point.
GroupHeight	Specifies the group's height.
Groups	Retrieves the control's groups collection.
HighlightItemType	Specifies the way how the control highlights the item.
HTMLPicture	Adds or replaces a picture in HTML captions.
hWnd	Retrieves the handle of the control's window.
Images	Sets the control's handle image list.
ItemFromPoint	Retrieves the item from point.
MarkSelectGroup	Specifies whether the selected group is marked using SelBackColorGroup and SelForeColorGroup properties.
Orientation	Specifies the control's orientation.
Picture	Retrieves or sets a graphic to be displayed in the control's background.
PictureDisplay	Retrieves or sets a value that indicates the way how the graphic is displayed on the control's background.
Replacelcon	Adds a new icon, replaces an icon or clears the control's image list.
SelBackColorGroup	Retrieves or sets a value that indicates the group's background color, if it's selected.
SelectGroup	Retrieves or sets a value that specifies the index of selected group.
SelectItemType	Retrieves or sets a value that indicates how the selected item is displayed.

SelectShortcut	Selects and displays the specified shortcut.
SelfForeColorGroup	Retrieves or sets a value that indicates the group's foreground color, if it's selected.
ShortcutBarBackColor	Retrieves or sets the shortcut bar's background color.
ShortcutBarHeight	Selects and displays the specified shortcut.
ShortcutBarSelBackColor	Retrieves or sets the background color for the selected icon in the shortcut bar.
ShortcutBarSelCaptionBackColor	Retrieves or sets the background color for selected shortcut when its entire caption is displayed.
ShortcutPicture	Specifies a custom-size picture assigned to a shortcut.
ShortcutPictureHeight	Specifies the height in pixels of the custom size picture being displayed in the shortcut bar.
ShortcutPictureWidth	Specifies the width in pixels of the custom size picture being displayed in the shortcut bar.
ShortcutResizeBackColor	Retrieves or sets the background color for the shortcut's resize bar.
ShowImageList	Retrieves or sets a value that indicates whether the image list window is visible or hidden.
ShowShortcutBar	Retrieves or sets a value that indicates whether the image shortcut bar is visible or hidden.
ShowToolTip	Shows the specified tooltip at given position.
SmallIcons	Retrieves or sets a value that indicates whether the control uses small icons or large icons.
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.

property ListBar.AllowResizeShortcutBar as Boolean

Specifies whether the user can resize the shorcut bar, to allow multiple shortcuts to be visible.

Type	Description
Boolean	A Boolean expression that indicates whether the user can resize the shortcut bar.

By default, the AllowResizeShortcutBar property is True. Use the AllowResizeShortcutBar property to hide the resize bar of the control's shortcut bar. Use the [ShowShortcutBar](#) property to show or hide the control's shortcut bar. The [ShortcutResizeBackColor](#) property changes the visual appearance of the resizing bar of the shortcut bar. Use the [ExpandShortcutCount](#) property to specify the number of shortcuts that display their full caption, else the first icon in the caption is displayed or the assigned picture is displayed.

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

Retrieves the identifier of the anchor from point.

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

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

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

```
Private Sub ListBar1_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
    With ListBar1
        .ShowToolTip .AnchorFromPoint(-1, -1)
    End With
End Sub
```

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

```
Private Sub AxListBar1_MouseMoveEvent(ByVal sender As System.Object, ByVal e As AxEXLISTBARLib._IListBarEvents_MouseMoveEvent) Handles AxListBar1.MouseMoveEvent
    With AxListBar1
        .ShowToolTip(.get_AnchorFromPoint(-1, -1))
    End With
End Sub
```

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

```
private void axListBar1_MouseMoveEvent(object sender,
AxEXLISTBARLib._IListBarEvents_MouseMoveEvent e)
{
    axListBar1.ShowToolTip(axListBar1.get_AnchorFromPoint(-1, -1));
}
```

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

```
void OnMouseMoveListBar1(short Button, short Shift, long X, long Y)
{
    COleVariant vtEmpty; V_VT( &vtEmpty ) = VT_ERROR;
    m_listBar.ShowToolTip( m_listBar.GetAnchorFromPoint( -1, -1 ), vtEmpty, vtEmpty,
vtEmpty );
}
```

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

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

with thisform
    With .ListBar1
        .ShowToolTip(.AnchorFromPoint(-1, -1))
    EndWith
endwith
```


property ListBar.Appearance as AppearanceEnum

Specifies the control's appearance.

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

Use the Appearance property to specify the control's border. Use the [GroupAppearance](#) property to specify the appearance for the groups. Use the [BackColorGroup](#) and [BackColorGroup2](#) properties to specify the background colors for the groups. Use the [BackColor](#) property to specify the control's background color. Use the [ForeColor](#) property to specify the control's foreground color. Use the [Font](#) property to specify the control's font. Use the [Image](#) property to assign a picture to an item. Use the [Image](#) property to assign a picture to a group.

method ListBar.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 ListBar1_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 ListBar.BackColor as Color

Retrieves or sets a value that indicates the control's background color.

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 [BackColorGroup](#) property to specify the default background color used to pain the groups captions. Use the [BackColor](#) property to specify the background color group's caption, Use the [BackColorList](#) property to specify the background color of the group's list. Use the [BackColor](#) property to specify the item's background color. Use the [Picture](#) property to specify the control's picture displayed on its background. Use the <bgcolor> built in HTML tag in the [Caption](#) property to define portions of text using a specified background color.

property ListBar.BackColorGroup as Color

Retrieves or sets a value that indicates the default group's background color.

Type	Description
Color	A color expression that indicates the group'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.

Use the BackColorGroup property to specify the background color for all groups. Use the [BackColorGroup2](#) property to specify the second background color when drawing the caption of group like a gradient. The BackColorGroup property changes only the background color for the captions. Use the [BackColor](#), [BackColorList](#) property to change the background color for a specific group. Use the [SelBackColorGroup](#) property retrieves or sets a value that indicates the group's background color, if it's selected.

For instance, the following VB sample changes the visual appearance for group headers. The [BackColorGroup](#) property indicates the indicates the default group's 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 BackColorGroup property to indicates the index of the skin that we want to use. The sample applies the skin "".

```
With ListBar1
    .VisualAppearance.Add 1, "D:\Temp\ExListBar.Help\tabdown1.ebn"
    .BackColorGroup = &H1000000
End With
```

The following C++ sample changes the visual appearance for group headers:

```
#include "Appearance.h"
m_listbar.GetVisualAppearance().Add( 1, COleVariant(
"D:\\Temp\\ExListBar.Help\\tabup1.ebn" ) );
m_listbar.SetBackColorGroup( 0x1000000 );
```

The following VB.NET sample changes the visual appearance for group headers:

```
With AxListBar1
```

```
.VisualAppearance.Add(1, "D:\Temp\ExListBar.Help\tabup1.ebn")  
.Template = "BackColorGroup = 16777216"  
End With
```

The following C# sample changes the visual appearance for group headers:

```
axListBar1.VisualAppearance.Add(1, "D:\\Temp\\ExListBar.Help\\tabup1.ebn");  
axListBar1.Template = "BackColorGroup = 16777216";
```

The following VFP sample changes the visual appearance for group headers:

```
With thisform.ListBar1  
.VisualAppearance.Add(1, "D:\Temp\ExListBar.Help\tabup1.ebn")  
.BackColorGroup = 16777216  
EndWith
```

where the 16777216 value represents 0x1000000 in hexadecimal.

property ListBar.BackColorGroup2 as Color

Specifies the color at the ending boundary line of the gradient group's caption.

Type	Description
Color	A color expression that specifies the color at the ending boundary line of the gradient group's caption.

Use the [BackColorGroup](#) and BackColorGroup2 properties to display the caption of the group using a gradient color. Use the [BackColor](#), [BackColorList](#) property to change the background color for a specific group. Use the [Font](#) property to change the control's font.



property ListBar.Background(Part as BackgroundPartEnum) as Color

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

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

The Background property specifies a background color or a visual appearance for specific parts in the control. If the Background property is 0, the control draws the part as default. Use the [Add](#) method to add new skins to the control. Use the [Remove](#) method to remove a specific skin from the control. Use the [Clear](#) method to remove all skins in the control. Use the [BeginUpdate](#) and [EndUpdate](#) methods to maintain performance while init the control



The following VB sample changes the visual appearance for the selected item and for the item from the cursor. The sample uses the "" skin for the item from the cursor, and the "" skin for the selected item. Use the [SelectItemType](#) property to specify whether the control marks the selected item. The [HighlightItemType](#) property indicates whether the item from the cursor is highlighted.

```
ListBar1.SelectItemType = exSelectPush
With ListBar1
    .VisualAppearance.Add &H30, "D:\Temp\ExListBar.Help\select.ebn"
    .VisualAppearance.Add &H40, "D:\Temp\ExListBar.Help\highlight.ebn"
    .Background(exSelectedItem) = &H30000000
    .Background(exHightlightItem) = &H40000000
End With
```


The following C++ sample changes the visual appearance for the selected item and for the item from the cursor:

```
#include "Appearance.h"
m_listbar.GetVisualAppearance().Add( 0x30, COleVariant(
"D:\\Temp\\ExListBar.Help\\select.ebn" ) );
m_listbar.GetVisualAppearance().Add( 0x40, COleVariant(
"D:\\Temp\\ExListBar.Help\\highlight.ebn" ) );
m_listbar.SetBackground( 4, 0x30000000 );
m_listbar.SetBackground( 5, 0x40000000 );
```

The following VB.NET sample changes the visual appearance for the selected item and for the item from the cursor:

```
With AxListBar1
    .VisualAppearance.Add(&H30, "D:\\Temp\\ExListBar.Help\\select.ebn")
    .VisualAppearance.Add(&H40, "D:\\Temp\\ExListBar.Help\\highlight.ebn")
    .set_Background(EXLISTBARLib.BackgroundPartEnum.exSelectItem, &H30000000)
    .set_Background(EXLISTBARLib.BackgroundPartEnum.exHightlightItem, &H40000000)
End With
```

The following C# sample changes the visual appearance for the selected item and for the item from the cursor:

```
axListBar1.VisualAppearance.Add(0x30, "D:\\Temp\\ExListBar.Help\\select.ebn");
axListBar1.VisualAppearance.Add(0x40, "D:\\Temp\\ExListBar.Help\\highlight.ebn");
axListBar1.set_Background(EXLISTBARLib.BackgroundPartEnum.exSelectItem,
0x30000000);
axListBar1.set_Background(EXLISTBARLib.BackgroundPartEnum.exHightlightItem,
0x40000000);
```

The following VFP sample changes the visual appearance for the selected item and for the item from the cursor:

```
with thisform.ListBar1
    .VisualAppearance.Add(48, "D:\\Temp\\ExListBar.Help\\select.ebn") && 0x30
    .VisualAppearance.Add(64, "D:\\Temp\\ExListBar.Help\\highlight.ebn") && 0x40
    .Background(4) = 805306368      && exSelectItem, 0x30000000
    .Background(5) = 1073741824    && exHightlightItem, 0x40000000
```


method ListBar.BeginUpdate ()

Maintains performance when items are added to the control one at a time.

Type

Description

Use the BeginUpdate and [EndUpdate](#) methods to maintain performance while adding multiple groups and items. Use the [Add](#) method to add a new group to the control. Use the [AddItem](#) method to add new items to the group.

The following VB sample adds a group and two items:

```
With ListBar1
    .BeginUpdate
    With .Groups.Add("Group 1")
        .AddItem "Item 1"
        .AddItem "Item 2"
    End With
    .EndUpdate
End With
```

The following C++ sample adds a group and two items:

```
#include "Item.h"
#include "Group.h"
#include "Groups.h"
m_listbar.BeginUpdate();
COleVariant vtMissing; V_VT( &vtMissing ) = VT_ERROR;
CGroups groups = m_listbar.GetGroups();
CGroup group = groups.Add("Group 1");
group.AddItem( "Item 1", vtMissing );
group.AddItem( "Item 2", vtMissing );
m_listbar.EndUpdate();
```

The following VB.NET sample adds a group and two items:

```
With AxListBar1
    .BeginUpdate()
    With .Groups.Add("Group 1")
        .AddItem("Item 1")
    End With
End With
```

```
.AddItem("Item 2")  
End With  
.EndUpdate()  
End With
```

The following C# sample adds a group and two items:

```
axListBar1.BeginUpdate();  
EXLISTBARLib.Group group = axListBar1.Groups.Add("Group 1");  
group.AddItem("Item 1", null);  
group.AddItem("Item 2", null);  
axListBar1.EndUpdate();
```

The following VFP sample adds a group and two items:

```
With thisform.ListBar1  
.BeginUpdate()  
With .Groups.Add("Group 1")  
.AddItem("Item 1")  
.AddItem("Item 2")  
EndWith  
.EndUpdate()  
EndWith
```

property ListBar.BorderHeight as Long

Specifies the border's height.

Type	Description
Long	A long expression that indicates the border's height, in pixels.

By default, the BorderHeight property is 2 pixels. Use the [BorderWidth](#) property to specify the width of the control's border. Use the BorderHeigth and BorderWidth properties to define the size of the groups inside the control's client area.

property ListBar.BorderWidth as Long

Specifies the border's width.

Type	Description
Long	A long expression that indicates the border's width.

By default, the BorderWidth property is 2 pixels. Use the [BorderHeight](#) property to specify the height of the control's border. Use the BorderHeigth and BorderWidth properties to define the size of the groups inside the control's client area.

property ListBar.DelayScroll as Long

Specifies the delay used when user selects a group.

Type	Description
Long	A long expression that indicates the delay used by control when a new group is selected.

By default, the DelayScroll property is 50. Use the DelayScroll property to stop scrolling groups when the selection is changed. Use the [BackColorList](#) property to specify the background color for the group's list. Use the [BackColor](#) property to specify the background color for the group's caption.

method ListBar.EndUpdate ()

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

Type

Description

Use the [BeginUpdate](#) and EndUpdate methods to maintain performance while adding multiple groups and items. Use the [Add](#) method to add a new group to the control. Use the [AddItem](#) method to add new items to the group.

The following VB sample adds a group and two items:

```
With ListBar1
    .BeginUpdate
    With .Groups.Add("Group 1")
        .AddItem "Item 1"
        .AddItem "Item 2"
    End With
    .EndUpdate
End With
```

The following C++ sample adds a group and two items:

```
#include "Item.h"
#include "Group.h"
#include "Groups.h"
m_listbar.BeginUpdate();
COleVariant vtMissing; V_VT( &vtMissing ) = VT_ERROR;
CGroups groups = m_listbar.GetGroups();
CGroup group = groups.Add("Group 1");
group.AddItem( "Item 1", vtMissing );
group.AddItem( "Item 2", vtMissing );
m_listbar.EndUpdate();
```

The following VB.NET sample adds a group and two items:

```
With AxListBar1
    .BeginUpdate()
    With .Groups.Add("Group 1")
        .AddItem("Item 1")
    End With
End With
```



```
.AddItem("Item 2")  
End With  
.EndUpdate()  
End With
```

The following C# sample adds a group and two items:

```
axListBar1.BeginUpdate();  
EXLISTBARLib.Group group = axListBar1.Groups.Add("Group 1");  
group.AddItem("Item 1", null);  
group.AddItem("Item 2", null);  
axListBar1.EndUpdate();
```

The following VFP sample adds a group and two items:

```
With thisform.ListBar1  
.BeginUpdate()  
With .Groups.Add("Group 1")  
.AddItem("Item 1")  
.AddItem("Item 2")  
EndWith  
.EndUpdate()  
EndWith
```

property ListBar.EventParam(Parameter as Long) as Variant

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

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

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

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

In most languages you will type something like:

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

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

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

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

reference.

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

method ListBar.ExecuteTemplate (Template as String)

Executes a template and returns the result.

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

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

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

```
Debug.Print ListBar1.ExecuteTemplate("BackColor")
```

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

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

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

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

An instruction can be one of the following:

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

The Template supports the following general functions:

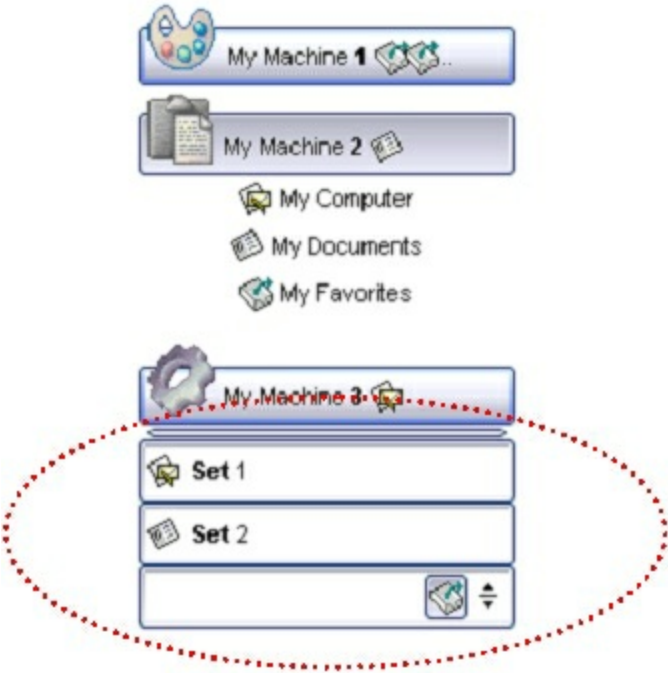
- **RGB(R,G,B)** *property retrieves an RGB value, where the R, G, B are byte values that indicates the R G B values for the color being specified. For instance, the following code changes the control's background color to red: BackColor = RGB(255,0,0)*
- **CreateObject(progID)** *property creates and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

property ListBar.ExpandShortcutCount as Long

Retrieves or sets a value that indicates the number of shortcuts being expanded.

Type	Description
Long	A long expression that indicates the number of shortcuts that display their full caption.

By default, the ExpandShortcutCount property is 0. The ExpandShortcutCount property is changed when the user resizes the shortcut bar. Use the [ShowShortcutBar](#) property to show or hide the control's shortcut bar. Use the [AllowResizeShortcutBar](#) property to enable or disable resizing the shortcut bar. Use the [ExpandShortcutImage](#) property to hide the expand button in the control's shortcut bar, or to change the icon of the expand button in the shortcut bar. The [Shortcut](#) property indicates the HTML caption of the shortcut that displays the specified group. Groups with the same Shortcut property are displayed in the same shortcut.



property ListBar.ExpandShortcutImage as Long

Retrieves or sets a value that indicates the index of the image being displayed to expand the shortcuts.

Type	Description
Long	A Long expression that indicates the index of the icon being used to display the expand button in the control's shortcut bar.

By default, The ExpandShortcutImage property is 0. If the ExpandShortcutImage property is 0, the control displays the default icon to show the expand button in the shortcut bar. If the ExpandShortcutImage property is greater than 0, it indicates the index of the icon being used to display the expand button. The [Images](#) method assigns a collection of icons to the control. The expand button in the shortcut bar is hidden, if the ExpandShortcutImage property is -1. Use the [AllowResizeShortcutBar](#) property to enable or disable resizing the shortcut bar. Use the [ExpandShortcutCount](#) property to specify the number of shortcuts being expanded. An expanded shortcut displays its full caption, as [Shortcut](#) property specifies.

property ListBar.Font as IFontDisp

Retrieves or sets the control's font.

Type	Description
IFontDisp	A Font object that specifies the control's font.

Use the Font property to specify the control's font. Use the [Bold](#), [Italic](#), [Underline](#) or [StrikeOut](#) property to define the font for caption of the group. Use the [Caption](#) property to define the caption of the group. Use the [Bold](#), [Italic](#), [Underline](#) or [StrikeOut](#) property to define the font for caption of the item. Use the [Caption](#) property to define the caption of the item. The [GroupHeight](#) property specifies the height of the caption for all groups, in pixels. Use the [ItemHeight](#) property to specify height of the items in the group's list.

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

```
With ListBar1
    With .Font
        .Name = "Tahoma"
    End With
End With
```

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

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

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

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

```
With AxListBar1
    Dim font As System.Drawing.Font = New System.Drawing.Font("Tahoma", 10,
    FontStyle.Regular, GraphicsUnit.Point)
    .Font = font
End With
```

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

```
System.Drawing.Font font = new System.Drawing.Font("Tahoma", 10, FontStyle.Regular);
axListBar1.Font = font;
```


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

```
with thisform.ListBar1.Object  
    .Font.Name = "Tahoma"  
endwith
```

The following Template sample changes by code a new font to the control:

```
Font  
{  
    Name = "Tahoma"  
}
```

property ListBar.ForeColor as Color

Retrieves or sets a value that indicates 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 [ForeColorGroup](#) property to specify the default foreground color used to paint the groups captions. Use the [ForeColor](#) property to specify the foreground color for group's caption, Use the [ForeColorList](#) property to specify the foreground color of the group's list. Use the [ForeColor](#) property to specify the item's foreground color. Use the <fgcolor> built in HTML tag in the [Caption](#) property to define portions of text using a specified foreground color.

property ListBar.ForeColorGroup as Color

Retrieves or sets a value that indicates the default group's foreground color.

Type	Description
Color	A color expression that indicates the default foreground color for group captions.

Use the ForeColorGroup property to specify the foreground color for all groups. The ForeColorGroup property changes only the foreground color of group captions. Use the [ForeColor](#), [ForeColorList](#) property to change the foreground color for a specific group. Use the <fgcolor> built in HTML tag in the [Caption](#) property to define portions of text using a specified foreground color.

property ListBar.FormatAnchor(New as Boolean) as String

Specifies the visual effect for anchor elements in HTML captions.

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

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

property ListBar.GroupAppearance as AppearanceEnum

Specifies the group's appearance.

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

Use the GroupAppearance to specify the group's appearance. Use the [BackColorGroup](#) and [BackColorGroup2](#) properties to specify the background colors for the groups. Use the [BackColor](#) property to specify the control's background color. Use the [ForeColor](#) property to specify the control's foreground color. Use the [Font](#) property to specify the control's font.

property ListBar.GroupFromPoint (X as OLE_XPOS_PIXELS, Y as OLE_YPOS_PIXELS) as Group

Gets the group from point.

Type	Description
X as OLE_XPOS_PIXELS	A single that specifies the current X location of the mouse pointer. The x values is always expressed in client coordinates
Y as OLE_YPOS_PIXELS	A single that specifies the current X location of the mouse pointer. The x values is always expressed in client coordinates
Group	A Group object over the point (X, Y).

Use the GroupFromPoint property to get the group over cursor. **If the X parameter is -1 and Y parameter is -1 the GroupFromPoint property determines the group from the cursor.** Use the [ItemFromPoint](#) property to retrieve the item over cursor. Use the [Caption](#) property to specify the caption of the group.

The following VB sample displays the caption of the group from the cursor:

```
Private Sub ListBar1_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
    With ListBar1
        Dim g As Group
        Set g = .GroupFromPoint(X / Screen.TwipsPerPixelX, Y / Screen.TwipsPerPixelY)
        If Not g Is Nothing Then
            Debug.Print g.Caption
        End If
    End With
End Sub
```

The following C++ sample displays the caption of the group from the cursor:

```
void OnMouseMoveListbar1(short Button, short Shift, long X, long Y)
{
    CGroup group = m_listbar.GetGroupFromPoint( X, Y );
    if ( group.m_lpDispatch != NULL )
        OutputDebugString( group.GetCaption() );
}
```

The following VB.NET sample displays the caption of the group from the cursor:

```
Private Sub AxListBar1_MouseMoveEvent(ByVal sender As Object, ByVal e As
AxEXLISTBARLib._IListBarEvents_MouseMoveEvent) Handles AxListBar1.MouseDownEvent
    With AxListBar1
        Dim g As EXLISTBARLib.Group = .get_GroupFromPoint(e.x, e.y)
        If Not g Is Nothing Then
            Debug.WriteLine(g.Caption)
        End If
    End With
End Sub
```

The following C# sample displays the caption of the group from the cursor:

```
private void axListBar1_MouseMoveEvent(object sender,
AxEXLISTBARLib._IListBarEvents_MouseMoveEvent e)
{
    EXLISTBARLib.Group g = axListBar1.get_GroupFromPoint(e.x, e.y);
    if (g != null)
        System.Diagnostics.Debug.WriteLine(g.Caption);
}
```

The following VFP sample displays the caption of the group from the cursor:

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

With thisform.ListBar1
    local g
    g = .GroupFromPoint(x , y)
    If !isnull(g)
        with g
            wait window nowait .Caption
        endwhile
    EndIf
EndWith
```

property ListBar.GroupHeight as Long

Specifies the group's height.

Type	Description
Long	A long expression that indicates the height of group's caption.

By default, the GroupHeight property is 22 pixels. The GroupHeight property specifies the height of the caption for all groups, in pixels. Use the [BorderHeight](#) and [BorderWidth](#) properties to define the size of the groups inside the control's client area. Use the [Font](#) property to specify the control's font. Use the [ItemHeight](#) property to specify height of the items in the group's list.

property ListBar.Groups as Groups

Retrieves the control's groups collection.

Type	Description
Groups	A Groups object that indicates the control's groups collection.

Use the Groups property to access the control's groups collection. Use the [Add](#) method property to add new groups to the control. Use the [AddItem](#) method to add new items to the group. Use the [BeginUpdate](#) and [EndUpdate](#) methods to maintain performance while adding new groups or new items. Use the [Font](#) property to specify the control's font. Use the [GroupHeight](#) property to specify the height of the captions for all groups. Use the [ItemHeight](#) property to specify height of the items in the group's list.

The following VB sample adds two groups and two items to each group:

```
With ListBar1
    .BeginUpdate
    With .Groups
        With .Add("Group 1")
            .AddItem "Item 1"
            .AddItem "Item 2"
        End With
        With .Add("Group 2")
            .AddItem "Item 1"
            .AddItem "Item 2"
        End With
    End With
    .EndUpdate
End With
```

The following C++ sample adds two groups and two items to each group:

```
#include "Item.h"
#include "Group.h"
#include "Groups.h"
COleVariant vtMissing; V_VT( &vtMissing ) = VT_ERROR;
m_listbar.BeginUpdate();
CGroups groups = m_listbar.GetGroups();
```

```

CGroup group1 = groups.Add( "Group 1" );
group1.AddItem( "Item 1", vtMissing );
group1.AddItem( "Item 2", vtMissing );
CGroup group2 = groups.Add( "Group 2" );
group2.AddItem( "Item 1", vtMissing );
group2.AddItem( "Item 2", vtMissing );
m_listbar.EndUpdate();

```

The following VB.NET sample adds two groups and two items to each group:

```

With AxListBar1
    .BeginUpdate()
    With .Groups
        With .Add("Group 1")
            .AddItem("Item 1")
            .AddItem("Item 2")
        End With
        With .Add("Group 2")
            .AddItem("Item 1")
            .AddItem("Item 2")
        End With
    End With
    .EndUpdate()
End With

```

The following C# sample adds two groups and two items to each group:

```

axListBar1.BeginUpdate();
EXLISTBARLib.Group group1 = axListBar1.Groups.Add("Group 1");
group1.AddItem("Item 1", null);
group1.AddItem("Item 2", null);
EXLISTBARLib.Group group2 = axListBar1.Groups.Add("Group 2");
group2.AddItem("Item 1", null);
group2.AddItem("Item 2", null);
axListBar1.EndUpdate();

```

The following VFP sample adds two groups and two items to each group:

```

With thisform.ListBar1

```

.BeginUpdate()

With .Groups

With .Add("Group 1")

.AddItem("Item 1")

.AddItem("Item 2")

EndWith

With .Add("Group 2")

.AddItem("Item 1")

.AddItem("Item 2")

EndWith

EndWith

.EndUpdate()

EndWith

property ListBar.HighlightItemType as HighlightItemEnum

Specifies the way how the control highlights the item.

Type	Description
HighlightItemEnum	A HighlightItemEnum expression that indicates the way how control marks the highlighted item.

Use the HighlightItemType property to specify the way how the control marks the highlighted item. If the HighlightItem property is exNoHighlight the item from the cursor is not marked at all. Use the [ForeColor](#) property to specify the item's foreground color. Use the [Underline](#) property to underline an item. Use the [ItemFromPoint](#) property to retrieve the item from point. The control fires the [HighlightItem](#) event when the cursor hovers an item. Use the [Background\(exHighlightItem\)](#) property to apply a skin for the item from the cursor.

property ListBar.HTMLPicture(Key as String) as Variant

Adds or replaces a picture in HTML captions.

Type	Description
Key as String	A String expression that indicates the key of the picture being added or replaced. If the Key property is Empty string, the entire collection of pictures is cleared.
Variant	<p>The HTMLPicture specifies the picture being associated to a key. It can be one of the followings:</p> <ul style="list-style-type: none">• a string expression that indicates the path to the picture file, being loaded.• a string expression that indicates the base64 encoded string that holds a picture object, Use the eximages tool to save your picture as base64 encoded format.• A Picture object that indicates the picture being added or replaced. (A Picture object implements IPicture interface), <p>If empty, the picture being associated to a key is removed. If the key already exists the new picture is replaced. If the key is not empty, and it doesn't not exist a new picture is added</p>

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

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

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

property ListBar.hWnd as Long

Retrieves the handle of the control's window.

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

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

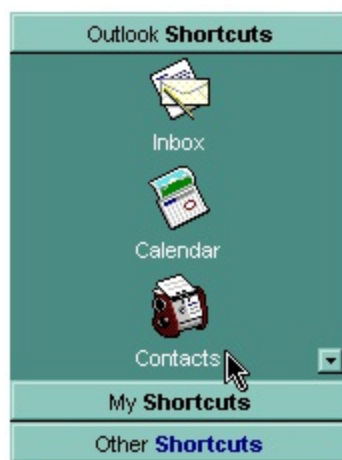
method ListBar.Images (Handle as Variant)

Sets the control's handle image list.

Type	Description
Handle as Variant	The Handle parameter can be:
	<ul style="list-style-type: none">• A string expression that specifies the ICO file to add. The ICO file format is an image file format for computer icons in Microsoft Windows. ICO files contain one or more small images at multiple sizes and color depths, such that they may be scaled appropriately. For instance, Images("c:\temp\copy.ico") method adds the sync.ico file to the control's Images collection (<i>string, loads the icon using its path</i>)
	<ul style="list-style-type: none">• A string expression that indicates the BASE64 encoded string that holds the icons list. Use the Exontrol's ExImages tool to save/load your icons as BASE64 encoded format. In this case the string may begin with "gBJJ..." (<i>string, loads icons using base64 encoded string</i>)
	<ul style="list-style-type: none">• A reference to a Microsoft ImageList control (mscomctl.ocx, MSComctlLib.ImageList type) that holds the icons to add (<i>object, loads icons from a Microsoft ImageList control</i>)
	<ul style="list-style-type: none">• A reference to a Picture (IPictureDisp implementation) that holds the icon to add. For instance, the VB's LoadPicture (Function LoadPicture([FileName], [Size], [ColorDepth], [X], [Y]) As IPictureDisp) or LoadResPicture (Function LoadResPicture(id, restype As Integer) As IPictureDisp) returns a picture object (<i>object, loads icon from a Picture object</i>)
	<ul style="list-style-type: none">• A long expression that identifies a handle to an Image List Control (the Handle should be of HIMAGELIST type). On 64-bit platforms, the Handle parameter must be a Variant of LongLong / LONG_PTR data type (signed 64-bit (8-byte) integers), saved under lVal field, as VT_I8 type. The LONGLONG / LONG_PTR is __int64, a 64-bit integer. For instance, in C++ you can use as Images(COleVariant((LONG_PTR)hImageList)) or Images(COleVariant((LONGLONG)hImageList)), where hImageList is of

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

The control provides an image list window, that's displayed at design time. Use the [ShowImageList](#) property to hide the image list window, at design time. At design time, the user can add new icons to the control's Images collection, by dragging icon files, exe files, etc, to the images list window. At runtime, the user can use the Images and [Replacelcon](#) method to change the Images collection. Use the [Image](#) property to assign a picture to a group. Use the [Image](#) property to assign a picture to an item. Use the [SmallIcons](#) property to specify the size of the icons being displayed. Use the **** HTML tag to insert icons inside the item's caption, if the [CaptionFormat](#) property is exHTML.



The following VB sample uses the Microsoft Image List control:

```
ListBar1.Images ImageList1.hImageList
```

The following VB sample loads a collection of icons from a BASE64 encoded string:

```
With ListBar1
  Dim s As String
  s =
  "gBJJgBAIEAAGAEGCAAhb/hz/EIAh8Tf5CJo2AEZjQAjEZFEaIEaEEaAIAkcbk0oIUrlktl0vmExml

  s = s +
  "FphZDEJtT1zp7bd1XasiLB8ld4o8kCRJIACSpRfNKWXd1uH+eaHn8P55necB/kmf5+A+eBnr

  .BeginUpdate
  .Images s
  With .Groups
```



```

With .Add("Group 1")
    .AddItem "Item 1", 1
    .AddItem "Item 2", 2
End With
With .Add("Group 2")
    .AddItem "Item 1", 2
    .AddItem "Item 2", 1
End With
End With
.EndUpdate
End With

```

The following C++ sample loads a collection of icons from a BASE64 encoded string:

```

#include "Item.h"
#include "Group.h"
#include "Groups.h"
COleVariant vtMissing; V_VT( &vtMissing ) = VT_ERROR;
CString
s("gBJJgBAIEAAGAEGCAAhb/hz/EIAh8Tf5CJo2AEZjQAjEZFEalEaEEaAlAkcbk0oIUrlktl0vmExr

s = s +
"FphZDEJtT1zp7bd1XasiLB8ld4o8kCRJIACSpRfNKWXd1uH+eaHn8P55necB/kmf5+A+eBnr

m_listbar.BeginUpdate();
CGroups groups = m_listbar.GetGroups();
m_listbar.Images(COleVariant(s));
CGroup group1 = groups.Add( "Group 1" );
group1.AddItem( "Item 1", COleVariant(long(1)) );
group1.AddItem( "Item 2", COleVariant(long(2)) );
CGroup group2 = groups.Add( "Group 2" );
group2.AddItem( "Item 1", COleVariant(long(3)) );
group2.AddItem( "Item 2", COleVariant(long(4)) );
m_listbar.EndUpdate();

```

The following VB.NET sample loads a collection of icons from a BASE64 encoded string:

```

With AxListBar1

```

```

Dim s As String =
"gBJJgBAIEAAGAEGCAAhb/hz/EIAh8Tf5CJo2AEZjQAjEZFEaIEaEEaAIAkcbk0oIUrlktl0vmExml

s = s +
"FphZDEJtT1zp7bd1XasiLB8ld4o8kCRJIACSpRfNKWXd1uH+eaHn8P55necB/kmf5+A+eBnr

.BeginUpdate()
.Images(s)
With .Groups
    With .Add("Group 1")
        .AddItem("Item 1", 1)
        .AddItem("Item 2", 2)
    End With
    With .Add("Group 2")
        .AddItem("Item 1", 3)
        .AddItem("Item 2", 4)
    End With
End With
.EndUpdate()
End With

```

The following C# sample loads a collection of icons from a BASE64 encoded string:

```

String s=
"gBJJgBAIEAAGAEGCAAhb/hz/EIAh8Tf5CJo2AEZjQAjEZFEaIEaEEaAIAkcbk0oIUrlktl0vmExml

s = s +
"FphZDEJtT1zp7bd1XasiLB8ld4o8kCRJIACSpRfNKWXd1uH+eaHn8P55necB/kmf5+A+eBnr

axListBar1.BeginUpdate();
axListBar1.Images(s);
EXLISTBARLib.Group group1 = axListBar1.Groups.Add("Group 1");
group1.AddItem("Item 1", 1);
group1.AddItem("Item 2", 2);
EXLISTBARLib.Group group2 = axListBar1.Groups.Add("Group 2");
group2.AddItem("Item 1", 3);
group2.AddItem("Item 2", 4);

```

```
axListBar1.EndUpdate();
```

The following VFP sample loads a collection of icons from a BASE64 encoded string:

```
With thisform.ListBar1
  local s
  .BeginUpdate()
  s =
"GBJJgBAIEAAJAEGCAAhb/hz/EIAh8Tf5CJo2AEZjQAjEZFEaIEaGEaAIAEEbjMjIErIktl0vmExmU

  s = s +
"Fw2HxGJxWLxmNx2PyGRyWTymVy2XzGZzWbzmdz2f0Gh0Wj0ml02npqAQEZ1WojWq1b.

  s = s +
"Gjb8wQ4L1us7zeQUmbqP29z4uu3rWv+ +jVwo7kKus3MNRA1cOQk6SgRPEkMwu3EXRY9'

  s = s +
"A/koxFKE6Q3O6WT9M8MSHQ0MuXQLpSXLE3xZJUwJy8bW0W+UPT7DEnUIMEa0xHdMC

  s = s +
"pdLzXZfsNX8+qfUEs2Bxg5IJwerN7Jld7LWlz2Atc0qLB8ImKo8kCRJIACSo2QGO11hOJM2hx

s = s + "+YJ/7YPaHnGiaAg="
With .Groups
  With .Add("Group 1")
    .AddItem("Item 1",1)
    .AddItem("Item 2",2)
  EndWith
  With .Add("Group 2")
    .AddItem("Item 1",3)
    .AddItem("Item 2",4)
  EndWith
EndWith
.EndUpdate()
EndWith
```

The following Template sample loads a collection of icons from a BASE64 encoded string:

BeginUpdate

Images("gBJJgBAIEAAJAEGCAAhb/hz/EIAh8Tf5CJo2AEZjQAjEZFEalEaGEaAIAEEbjMjlErlktlOv

Groups

```
{
  "Group 1"
  {
    AddItem("Item 1",1)
    AddItem("Item 2",2)
  }
  "Group 2"
  {
    AddItem("Item 1",3)
    AddItem("Item 2",4)
  }
}
```

EndUpdate

property ListBar.ItemFromPoint (X as OLE_XPOS_PIXELS, Y as OLE_YPOS_PIXELS) as Item

Retrieves the item from point.

Type	Description
X as OLE_XPOS_PIXELS	A single that specifies the current X location of the mouse pointer. The x values is always expressed in client coordinates
Y as OLE_YPOS_PIXELS	A single that specifies the current X location of the mouse pointer. The x values is always expressed in client coordinates
Item	An Item object over the point(X, Y).

Use the ItemFromPoint property to get the item from the cursor. **If the X parameter is -1 and Y parameter is -1 the GroupFromPoint property determines the item from the cursor.** Use the [GroupFromPoint](#) property to retrieve the group over cursor. Use the [Caption](#) property to specify the caption of the item.

The following VB sample displays the caption of the item from the cursor:

```
Private Sub ListBar1_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
    With ListBar1
        Dim i As Item
        Set i = .ItemFromPoint(X / Screen.TwipsPerPixelX, Y / Screen.TwipsPerPixelY)
        If Not i Is Nothing Then
            Debug.Print i.Caption
        End If
    End With
End Sub
```

The following C++ sample displays the caption of the item from the cursor:

```
void OnMouseMoveListbar1(short Button, short Shift, long X, long Y)
{
    Cltem item = m_listbar.GetItemFromPoint( X, Y );
    if ( item.m_lpDispatch != NULL )
        OutputDebugString( item.GetCaption() );
}
```

The following VB.NET sample displays the caption of the item from the cursor:

```
Private Sub AxListBar1_MouseMoveEvent(ByVal sender As Object, ByVal e As
AxEXLISTBARLib._IListBarEvents_MouseMoveEvent) Handles AxListBar1.MouseDownEvent
    With AxListBar1
        Dim i As EXLISTBARLib.Item = .get_ItemFromPoint(e.x, e.y)
        If Not i Is Nothing Then
            Debug.WriteLine(i.Caption)
        End If
    End With
End Sub
```

The following C# sample displays the caption of the item from the cursor:

```
private void axListBar1_MouseMoveEvent(object sender,
AxEXLISTBARLib._IListBarEvents_MouseMoveEvent e)
{
    EXLISTBARLib.Item i = axListBar1.get_ItemFromPoint(e.x, e.y);
    if (i != null)
        System.Diagnostics.Debug.WriteLine(i.Caption);
}
```

The following VFP sample displays the caption of the item from the cursor:

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

With thisform.ListBar1
    local i
    i = .ItemFromPoint(x , y)
    If !isnull(i)
        with i
            wait window nowait .Caption
        endwith
    EndIf
EndWith
```


property ListBar.MarkSelectGroup as Boolean

Specifies whether the selected group is marked using SelBackColorGroup and SelForeColorGroup properties.

Type	Description
Boolean	A boolean expression that indicates whether the control uses the SelBackColorGroup and SelForeColorGroup properties to mark the selected group's caption.

By default, the MarkSelectGroup property is False. If the MarkSelectGroup property is True, the control uses the [SelBackColorGroup](#) and [SelForeColorGroup](#) properties to display the selected group. Use the [BackColor](#) property to specify the group's background color. Use the [BackColorList](#) property to specify the background color for group's list. Use the [BackColorGroup](#) property to specify a default background color for groups.

property ListBar.Orientation as OrientationEnum

Specifies the control's orientation.

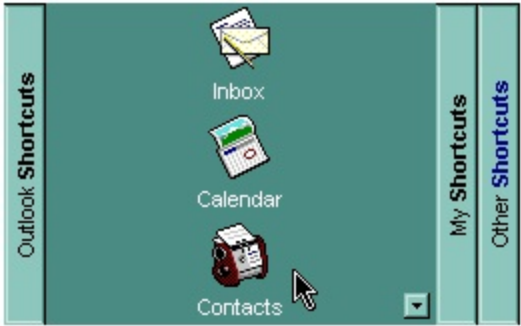
Type	Description
OrientationEnum	An OrientationEnum expression that indicates the control's orientation.

By default, the control's orientation is exVertical. The Orientation feature is supported Windows NT, Windows 2000, or Windows XP systems. The Orientation feature is not supported on Windows 95, Windows 98 or Windows Me systems.

The following screen shot shows the control when the Orientation property is exVertical:



The following screen shot shows the control when the Orientation property is exHorizontal:



property ListBar.Picture as IPictureDisp

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

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

The control's picture is displayed on the control's background. Use the [PictureDisplay](#) property to specify the way how the picture is arranged on the control's background. Use the [Picture](#) property to specify a picture for a given group. Use the [BackColor](#) property to specify the control's background. Use the [BackColor](#) property to specify the background color group's caption, Use the [BackColorList](#) property to specify the background color of the group's list. Use the [BackColor](#) property to specify the item's background color. Use the [DelayScroll](#) property to stop scrolling the groups when expanding or collapsing them.

property ListBar.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 arranged on the control's background.

The control's picture is displayed on the control's background. Use the PictureDisplay property to specify the way how the picture is arranged on the control's background. Use the [Picture](#) property to specify a picture for a given group. Use the [BackColor](#) property to specify the control's background. Use the [BackColor](#) property to specify the background color group's caption, Use the [BackColorList](#) property to specify the background color of the group's list. Use the [BackColor](#) property to specify the item's background color. Use the [DelayScroll](#) property to stop scrolling the groups when expanding or collapsing them.

method ListBar.Replacelcon ([Icon as Variant], [Index as Variant])

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

Type	Description
Icon as Variant	<p>A Variant expression that specifies the icon to add or insert, as one of the following options:</p> <ul style="list-style-type: none">• a long expression that specifies the handle of the icon (HICON)• a string expression that indicates the path to the picture file• a string expression that defines the picture's content encoded as BASE64 strings using the eXImages tool• a Picture reference, which is an object that holds image data. It is often used in controls like PictureBox, Image, or in custom controls (e.g., IPicture, IPictureDisp) <p>If the Icon parameter is 0, it specifies that the icon at the given Index is removed. Furthermore, setting the Index parameter to -1 removes all icons.</p> <p>By default, if the Icon parameter is not specified or is missing, a value of 0 is used.</p>
Index as Variant	<p>A long expression that defines the index of the icon to insert or remove, as follows:</p> <ul style="list-style-type: none">• A zero or positive value specifies the index of the icon to insert (when Icon is non-zero) or to remove (when the Icon parameter is zero)• A negative value clears all icons when the Icon parameter is zero <p>By default, if the Index parameter is not specified or is missing, a value of -1 is used.</p>
Return	Description
Long	A long expression that indicates the index of the icon in the images collection

Use the Replacelcon property to add, remove or replace an icon in the control's images

collection. Also, the `Replacelcon` property can clear the images collection. Use the `Images` method to attach an image list to the control. Use the [SmallIcons](#) property to select the icon's size. Use the [Images](#) method to attach an image list to the control.

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

`i = ListBar1.Replacelcon(LoadPicture("d:\icons\help.ico").Handle)`, where `i` is the index to insert the icon

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

`i = ListBar1.Replacelcon(LoadPicture("d:\icons\help.ico").Handle, 0)`, in this case the `i` is zero, because the first icon was replaced.

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

`ListBar1.Replacelcon 0, i`, in this case the `i` must be the index of the icon that follows to be removed

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


`ListBar1.Replacelcon 0, -1`

property ListBar.SelBackColorGroup as Color

Retrieves or sets a value that indicates the group's background color, if it's selected.

Type	Description
Color	A color expression that indicates the selected group'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.

The SelBackColorGroup property has effect only if [MarkSelectGroup](#) property is True. Use the SelBackColorGroup and [SelfForeColorGroup](#) properties to customize colors for selected group. Use the [BackColor](#) property to specify the group's background color. Use the [BackColorList](#) property to specify the background color for group's list. Use the [BackColorGroup](#) property to specify a default background color for groups.

For instance, the following VB sample changes the visual appearance for group headers. The [BackColorGroup](#) property indicates the indicates the default group's 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 BackColorGroup property to indicates the index of the skin that we want to use. The sample applies the skin "".

```
With ListBar1
    .VisualAppearance.Add 1, "D:\Temp\ExListBar.Help\tabdown1.ebn"
    .BackColorGroup = &H1000000
End With
```

The following C++ sample changes the visual appearance for group headers:

```
#include "Appearance.h"
m_listbar.GetVisualAppearance().Add( 1, COleVariant(
"D:\\Temp\\ExListBar.Help\\tabup1.ebn" ) );
m_listbar.SetBackColorGroup( 0x1000000 );
```

The following VB.NET sample changes the visual appearance for group headers:

```
With AxListBar1
```

```
.VisualAppearance.Add(1, "D:\Temp\ExListBar.Help\tabup1.ebn")  
.Template = "BackColorGroup = 16777216"  
End With
```

The following C# sample changes the visual appearance for group headers:

```
axListBar1.VisualAppearance.Add(1, "D:\\Temp\\ExListBar.Help\\tabup1.ebn");  
axListBar1.Template = "BackColorGroup = 16777216";
```

The following VFP sample changes the visual appearance for group headers:

```
With thisform.ListBar1  
.VisualAppearance.Add(1, "D:\Temp\ExListBar.Help\tabup1.ebn")  
.BackColorGroup = 16777216  
EndWith
```

where the 16777216 value represents 0x1000000 in hexadecimal.

property ListBar.SelectGroup as Long

Retrieves or sets a value that specifies the index of selected group.

Type	Description
Long	A long expression that indicates the index of the selected group.

Use the SelectGroup property to select a group. The [SelectGroup](#) event is fired when a new group is selected. Use the [SelectItem](#) property to retrieves the index of selected item. Use the [Caption](#) property to get the caption of the item. Use the [Caption](#) property to get the caption of the group.

property ListBar.SelectItemType as SelectItemEnum

Retrieves or sets a value that indicates how the selected item is displayed.

Type	Description
SelectItemEnum	A SelectItemEnum expression that indicates the way how control marks the selected item.

By default, the SelectItemType property is exSelectPush. Use the SelectItemType property to determine the way how the control marks the selected item. Use the [SelectItem](#) property to get the index of selected item. Use the [SelectGroup](#) property to retrieve the index of selected group. Use the [Background](#) property to specify a background color or a visual appearance for the selected or highlighted node. Use the [Background](#)(exSelectItem) property to apply a skin to the selected item.

property ListBar.SelectShortcut as Variant

Selects and displays the specified shortcut.

Type	Description
Variant	A String expression that indicates the caption of the Shortcut being selected.

The SelectShortcut property indicates the shortcut being selected. The [Shortcut](#) property of the Group object indicates the shortcuts that may be selected. The Group objects with the same Shortcut property indicates a set of groups, that may be selected using the SelectShortcut property. The [SelectShortcut](#) event is fired when the user clicks a shortcut or when the user calls the SelectShortcut property. The [ShowShortcutBar](#) property shows or hides the control's shortcut bar. The [ShortcutPicture](#) property assigns a custom size picture to a shortcut. Use the [ShortcutPictureWidth](#) and [ShortcutPictureHeight](#) properties to indicate the size of the picture being displayed in the shortcut bar. The [ShortcutBarHeight](#) property sets or gets a value that indicates the height in pixels of the control's shortcut bar.

property ListBar.SelForeColorGroup as Color

Retrieves or sets a value that indicates the group's foreground color, if it's selected.

Type	Description
Color	A color expression that indicates the foreground color for selected group.

The SelForeColorGroup property has effect only if [MarkSelectGroup](#) property is True. Use the [ForeColor](#) property to specify the foreground color for the group's caption. Use the [ForeColorList](#) property to specify the foreground color of the group's list. Use the [ForeColorGroup](#) property to specify the default foreground color. Use the <fgcolor> built in HTML tag in the [Caption](#) property to define portions of text using a specified foreground color.

property ListBar.ShortcutBarBackColor as Color

Retrieves or sets the shortcut bar's background color.

Type	Description
Color	A color expression that indicates the shortcut'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.

Use the ShortcutBarBackColor property to specify the background color of the control's shortcut bar. The visual appearance of the items in the shortcut bar can be defined using skins. The [ShortcutBarSelBackColor](#) property to specify the visual appearance/background color for the selected icon/picture in the last item of the shortcut bar. The last item in the shortcut bar displays only icons or custom size pictures for each shortcut defined using the [Shortcut](#) property of the Group object. The [ShortcutBarSelCaptionBackColor](#) property defines the visual appearance/background color for the selected shortcut, when the full HTML caption is displayed. The [ShortcutResizeBackColor](#) property defines the visual appearance/background color slider that resizes the shortcut bar.

property ListBar.ShortcutBarHeight as Long

Selects and displays the specified shortcut.

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

By default, the ShortcutBarHeight property is 24 pixels. The ShortcutBarHeight property defines the height for each item in the shortcut bar. For instance, if the shortcut bar has no expanded shortcuts, the ShortcutBarHeight property defines the height of the shortcut bar, that displays a single item where all shortcuts are displayed. Use the [ShowShortcutBar](#) property to show or hide the control's shortcut bar. The [ShortcutPicture](#) property assigns a custom size picture to a shortcut. Use the [ShortcutPictureWidth](#) and [ShortcutPictureHeight](#) properties to indicate the size of the picture being displayed in the shortcut bar.

property ListBar.ShortcutBarSelBackColor as Color

Retrieves or sets the background color for the selected icon in the shortcut bar.

Type	Description
Color	A color expression that indicates the background color for the selected shortcut. 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 ShortcutBarSelBackColor property to specify the visual appearance/background color for the selected icon/picture in the last item of the shortcut bar. The last item in the shortcut bar displays only icons or custom size pictures for each shortcut defined using the [Shortcut](#) property of the Group object. The [ShortcutBarSelCaptionBackColor](#) property defines the visual appearance/background color for the selected shortcut, when the full HTML caption is displayed. Use the [ShortcutBarBackColor](#) property to specify the background color of the control's shortcut bar. The visual appearance of the items in the shortcut bar can be defined using skins. The [ShortcutResizeBackColor](#) property defines the visual appearance/background color slider that resizes the shortcut bar.

property ListBar.ShortcutBarSelCaptionBackColor as Color

Retrieves or sets the background color for selected shortcut when its entire caption is displayed.

Type	Description
Color	A color expression that indicates the background color for the selected shortcut. 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.

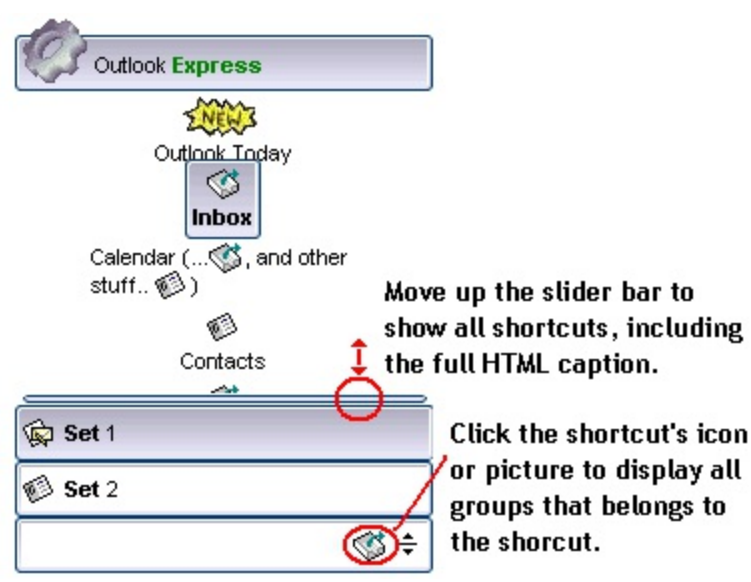
The ShortcutBarSelCaptionBackColor property defines the visual appearance/background color for the selected shortcut, when the full HTML caption is displayed. The [ShortcutBarSelBackColor](#) property to specify the visual appearance/background color for the selected icon/picture in the last item of the shortcut bar. The last item in the shortcut bar displays only icons or custom size pictures for each shortcut defined using the [Shortcut](#) property of the Group object. Use the [ShortcutBarBackColor](#) property to specify the background color of the control's shortcut bar. The visual appearance of the items in the shortcut bar can be defined using skins. The [ShortcutResizeBackColor](#) property defines the visual appearance/background color slider that resizes the shortcut bar.

property ListBar.ShortcutPicture(Shortcut as String) as Variant

Specifies a custom-size picture assigned to a shortcut.

Type	Description
Shortcut as String	A String expression that indicates the caption of the shortcut where a custom size picture is assigned.
Variant	A String expression that indicates the path to the picture file or a string expression that indicates the base64 encoded string that holds a picture object. Use the eximages tool to save your picture as base64 encoded format. A Picture object being assigned to the shortcut.

Use the ShorcutPicture property to assign a custom size picture to a shortcut. The [ShowShortcutBar](#) property specifies whether the control's shortcut bar is visible or hidden. The [Shortcut](#) property indicates the HTML caption of the shortcut that displays the specified group. Groups with the same Shortcut property are displayed in the same shortcut. The shortcut bar displays the first icon found in the HTML caption, if a custom size picture is not assigned to the shortcut using the ShortcutPicture property. Use the [ShortcutPictureWidth](#) and [ShortcutPictureHeight](#) properties to indicate the size of the picture being displayed in the shortcut bar. Use the [ShortcutBarHeight](#) property to define the height in pixels of one shortcut items in the shortcut bar. Use the [ExpandShortcutCount](#) property to expand the number of shortcuts in the control's shortcut bar. The ShorcutPicture property has no effect if the shortcut bar is not visible, or there is no Group assigned to the specified shortcut.



The following VB sample assign a custom size picture to the shortcut named "1 Set 1":

```
With ListBar1
    .ShortcutBarHeight = 38
```


.ShortcutPicture(" 1 Set 1") = "D:\Temp\Icons\misc.gif"
End With

The following screen shot shows the shortcutbar when there is no items expanded:



The following screen shot shows the shortcutbar when there is a single shortcut expanded (**Set 1**)



property ListBar.ShortcutPictureHeight as Long

Specifies the height in pixels of the custom size picture being displayed in the shortcut bar.

Type	Description
Long	A Long expression that indicates the height of the picture to be stretched to.

By default, the ShortcutPictureHeight property is -1. If the ShortcutPictureHeight property is -1, the shortcut's picture is not stretched on the height. If the ShortcutPictureHeight property is positive, it indicates the height in pixels to be stretched to. The ShortcutPictureHeight property specifies the height of the picture when assigning a custom size picture using the [ShortcutPicture](#) property. The [ShortcutPictureWidth](#) property specifies the width in pixels of the picture to be stretched to. Use the [ShortcutBarHeight](#) property to specify the height in pixels of the control's shortcut bar. The [ShowShortcutBar](#) property specifies whether the control's shortcut bar is visible or hidden. The [Shortcut](#) property indicates the HTML caption of the shortcut that displays the specified group. Groups with the same Shortcut property are displayed in the same shortcut. The shortcut bar displays the first icon found in the HTML caption, if a custom size picture is not assigned to the shortcut using the ShortcutPicture property.

property ListBar.ShortcutPictureWidth as Long

Specifies the width in pixels of the custom size picture being displayed in the shortcut bar.

Type	Description
Long	A long expression that indicates the width of the picture to be stretched to.

By default, the ShortcutPictureWidth property is -1. If the ShortcutPictureWidth property is -1, the shortcut's picture is not stretched on the width. If the ShortcutPictureWidth property is positive, it indicates the width in pixels to be stretched to. The ShortcutPictureWidth property specifies the width of the picture when assigning a custom size picture using the [ShortcutPicture](#) property. The [ShortcutPictureHeight](#) property specifies the height in pixels of the picture to be stretched to. The [ShowShortcutBar](#) property specifies whether the control's shortcut bar is visible or hidden. The [Shortcut](#) property indicates the HTML caption of the shortcut that displays the specified group. Groups with the same Shortcut property are displayed in the same shortcut. The shortcut bar displays the first icon found in the HTML caption, if a custom size picture is not assigned to the shortcut using the ShortcutPicture property.

property ListBar.ShortcutResizeBackColor as Color

Retrieves or sets the background color for the shortcut's resize bar.

Type	Description
Color	A color expression that indicates the background color for the slider that resizes the shortcut bar. 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 ShortcutResizeBackColor property defines the visual appearance/background color slider that resizes the shortcut bar. Use the [ShortcutBarBackColor](#) property to specify the background color of the control's shortcut bar. The visual appearance of the items in the shortcut bar can be defined using skins. The [ShortcutBarSelBackColor](#) property to specify the visual appearance/background color for the selected icon/picture in the last item of the shortcut bar. The last item in the shortcut bar displays only icons or custom size pictures for each shortcut defined using the [Shortcut](#) property of the Group object. The [ShortcutBarSelCaptionBackColor](#) property defines the visual appearance/background color for the selected shortcut, when the full HTML caption is displayed.

property ListBar.ShowImageList as Boolean

Retrieves or sets a value that indicates whether the image list window is visible or hidden.

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

The ShowImageList control has no effect at runtime. It has effect only at design time. Use the method to change the control's images list collection, or use [Replacelcon](#) method to add, remove, or clear the images collection. Use the [SmallIcons](#) property to select the icon's size. Use the [Images](#) method to assign a list of images to the control.

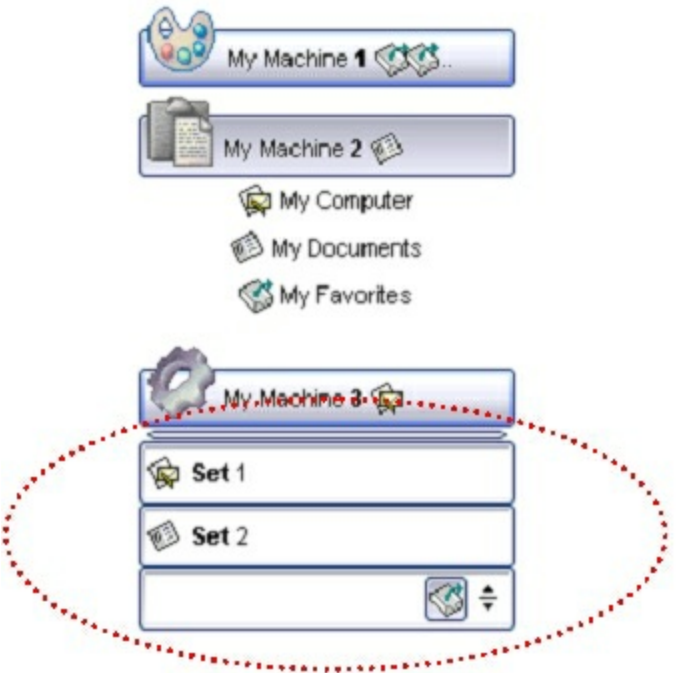


property ListBar.ShowShortcutBar as Boolean

Retrieves or sets a value that indicates whether the image shortcut bar is visible or hidden.

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

By default, the ShowShortcutBar property is False, and that means that the shortcut bar is hidden. The shortcut bar if visible, it is displayed on the bottom side of the control as seen in the following screen shot. The Shortcut feature allows you to group the groups in sets, so you may have sets that contains groups, and groups that contains items. The [Shortcut](#) property indicates the HTML caption of the shortcut that displays the specified group. Groups with the same Shortcut property are displayed in the same shortcut. The [ShortcutPicture](#) property assigns a custom size picture to a shortcut. Use the [ShortcutPictureWidth](#) and [ShortcutPictureHeight](#) properties to indicate the size of the picture being displayed in the shortcut bar. The [ShortcutBarHeight](#) property sets or gets a value that indicates the height in pixels of the control's shortcut bar. The [ShortcutBarBackColor](#) property indicates the shortcut bar's background color or its visual appearance if using skins. The [ShortcutResizeBackColor](#) property changes the visual appearance of the resizing bar of the shortcut bar. The [SelectShortcut](#) property selects a shortcut. When a shortcut is selected, the control displays only groups with the [Shortcut](#) property as being the SelectShortcut property.



The red circle marks the control's shortcut bar. Use the [AllowResizeShortcutBar](#) property to specify whether the user may expand the shortcut bar using the mouse. Use the [ExpandShortcutImage](#) property to display a custom icon in the right side expand button. Use the [ExpandShortcutCount](#) property to specify the number of shortcuts that display their

full caption, else the first icon in the caption is displayed or the assigned picture is displayed.

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

Shows the specified tooltip at given position.

Type	Description
ToolTip as String	<p>The ToolTip parameter can be any of the following:</p> <ul style="list-style-type: none">• NULL(BSTR) or "<null>"(string) to indicate that the tooltip for the object being hovered is not changed• A String expression that indicates the description of the tooltip, that supports built-in HTML format (adds, replaces or changes the object's tooltip)
Title as Variant	<p>The Title parameter can be any of the following:</p> <ul style="list-style-type: none">• missing (VT_EMPTY, VT_ERROR type) or "<null>" (string) the title for the object being hovered is not changed.• A String expression that indicates the title of the tooltip (no built-in HTML format) (adds, replaces or changes the object's title)
Alignment as Variant	<p>A long expression that indicates the alignment of the tooltip relative to the position of the cursor. If missing (VT_EMPTY, VT_ERROR) the alignment of the tooltip for the object being hovered is not changed.</p> <p>The Alignment parameter can be one of the following:</p> <ul style="list-style-type: none">• 0 - exTopLeft• 1 - exTopRight• 2 - exBottomLeft• 3 - exBottomRight• 0x10 - exCenter• 0x11 - exCenterLeft• 0x12 - exCenterRight• 0x13 - exCenterTop• 0x14 - exCenterBottom <p>By default, the tooltip is aligned relative to the top-left corner (0 - exTopLeft).</p>

Specifies the horizontal position to display the tooltip as one of the following:

- missing (VT_EMPTY, VT_ERROR type), indicates that the tooltip is shown on its default position / current cursor position (ignored)
- -1, indicates the current horizontal position of the cursor (current x-position)
- a numeric expression that indicates the horizontal screen position to show the tooltip (fixed screen x-position)
- a string expression that indicates the horizontal displacement relative to default position to show the tooltip (moved)

X as Variant

Specifies the vertical position to display the tooltip as one of the following:

- missing (VT_EMPTY, VT_ERROR type), indicates that the tooltip is shown on its default position / current cursor position (ignored)
- -1, indicates the current vertical position of the cursor (current y-position)
- a numeric expression that indicates the vertical screen position to show the tooltip (fixed screen y-position)
- a string expression that indicates the vertical displacement relative to default position to show the tooltip (displacement)

Y as Variant

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

For instance:

- [ShowToolTip\(<null>, <null>, , +8, +8\)](#), shows the tooltip of the object moved relative

to its default position

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

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

- ` bold ` bolds a part of the caption.
- `<u> underline </u>` specifies that the portion should appear as underlined.
- `<s> strikethrough </s>` specifies that the portion should appear as strikethrough.
- `<i> italic </i>` specifies that the portion should appear as italic.
- `<fgcolor=FF0000> fgcolor </fgcolor>` changes the foreground color for a portion.
- `<bgcolor=FF0000> bgcolor </bgcolor>` changes the background color for a portion.
- `
` breaks a line.
- `<solidline>` draws a solid line. If has no effect for a single line caption.
- `<dottedline>` draws a dotted line. If has no effect for a single line caption.
- `<upline>` draws the line to the top of the text line
- `<r>` aligns the rest of the text line to the right side. It has no effect if the caption contains a single line.
- `number[:width]` inserts an icon inside the cell's caption. The number indicates the index of the icon being inserted. Use the [Images](#) method to assign a list of icons to your chart. The last 7 bits in the high significant byte of the number expression indicates the identifier of the skin being used to paint the object. Use the [Add](#) method to add new skins to the control. If you need to remove the skin appearance from a part of the control you need to reset the last 7 bits in the high significant byte of the color being applied to the part. The width is optional and indicates the width of the icon being inserted. Using the width option you can overwrite multiple icons getting a nice effect. By default, if the width field is missing, the width is 18 pixels.
- `key[:width]` inserts a custom size picture being 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.
- `text ` displays portions of text with a different font and/or different size. For instance, the `bit` draws the bit text using the Tahoma font, on size 12 pt. If the name of the font is missing, and instead size is present, the current font is used with a different size. For instance, `bit` displays the bit text using the current font, but with a different size.
- `&` glyph characters as `&` (&), `<` (<), `>` (>), `"` ("), `&#number`, For

instance, the `€` displays the EUR character, in UNICODE configuration. The `&` ampersand is only recognized as markup when it is followed by a known letter or a `#` character and a digit. For instance if you want to display `bold` in HTML caption you can use `bold`;

Also, newer HTML format supports decorative text like follows:

- **`<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 `rrggb`, `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 FFFFFFFF;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 ListBar.SmallIcons as Boolean

Retrieves or sets a value that indicates whether the control uses small icons or large icons.

Type	Description
Boolean	A boolean expression that indicates whether the control uses small icons or large icons.

By default, the SmallIcons property is True. Use the SmallIcons property to specify the size of icons being used. If the SmallIcons property is True, the control displays 16x16 size icons. If the SmallIcons property is False, the control displays 32x32 size icons. Use the [Image](#) property to assign an icon or a custom size picture to the item. Use the [Image](#) property to assign an icon or a custom size picture to the group. Use the [Images](#) property to assign an image list to the control. Use the [GroupHeight](#) property to specify the height of the captions for all groups. Use the [ItemHeight](#) property to specify the height of the items inside the group. Use the [ShowImageList](#) property to show the control's images list at runtime.

The following screen shot displays 16x16 icons:



The following screen shot displays 32x32 icons:



property ListBar.Template as String

Specifies the control's template.

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

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

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*

For instance, the following template string adds two groups, and an item to each group:

```
BeginUpdate
Groups
{
    "Group 1"
    {
        AddItem("Item 1",1)
        Expanded = True
    }
    "Group 2"
    {
        AddItem("Item 1",2)
    }
}
EndUpdate
```

The control's editor looks like follows:

Property Pages

Color

Picture

Font

Template

The template editor helps you to create template files using a simple X-Script, that combines the XML style with a language close to VBScript. The editor automatically updates the control's look and feel while you are editing the template file.

http://www.exontrol.com

Outlook Shortcuts

My Shortcuts

Draft

Outbox

Sent Items

Journal

Outlook Update

Other Shortcuts

Help

New

Refresh

' Properties, Objects, Methods

' Comments

DelayScroll = 200

' Specifies the scroll delay. When the u:

ForeColor = 2147483653

' Changes the control's foreground colo

BackColor = 2147483660

' Changes the control's backgroun

BackColorGroup = 2147483652

' Applies gradient color for groups

BackColorGroup2 = 2147483660

' Applies gradient color for groups capti

SelectItemType = 0

' Gets the groups collection

Groups

' Opens the groups context

{

' Appenc

"Outlook <fgcolor=008000>Shortcuts"

' Opens the group context

{

' Assign an icon to the group

CaptionFormat = 1

' Specifies the item's height

Image = 1

' Specifies the item's height

ItemHeight = 36

' Appends a new item to group object

AddItem("Outlook Today")

' Opens the item context

{

Tip. You can invoke the control's list of properties and methods by pressing the CTRL + SPACE key. While you have it opened, keep the mouse over an property and you have the property's description also.

OK

Cancel

Apply

property ListBar.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.Active1.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 ListBar.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 use constant expressions as follows:

- *boolean* expression with possible values as *True* or *False*
- *numeric* expression may start with 0x which indicates a hexa decimal representation, else it should start with a 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 indicate 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 ListBar.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. Use the [ToolTip](#) property to specify the tooltip to be shown when the cursor hovers its caption. Use the [ShowToolTip](#) method to display a custom tooltip.

property ListBar.ToolTipFont as IFontDisp

Retrieves or sets the tooltip's font.

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

Use the ToolTipFont property to assign a font for the control's tooltip. The [ToolTipPopDelay](#) property specifies the period in ms of time the ToolTip remains visible if the mouse pointer is stationary within a control. Use the [ToolTipWidth](#) property to specify the width of the tooltip window. Use the [ToolTip](#) property to specify the tooltip to be shown when the cursor hovers its caption.

property ListBar.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.

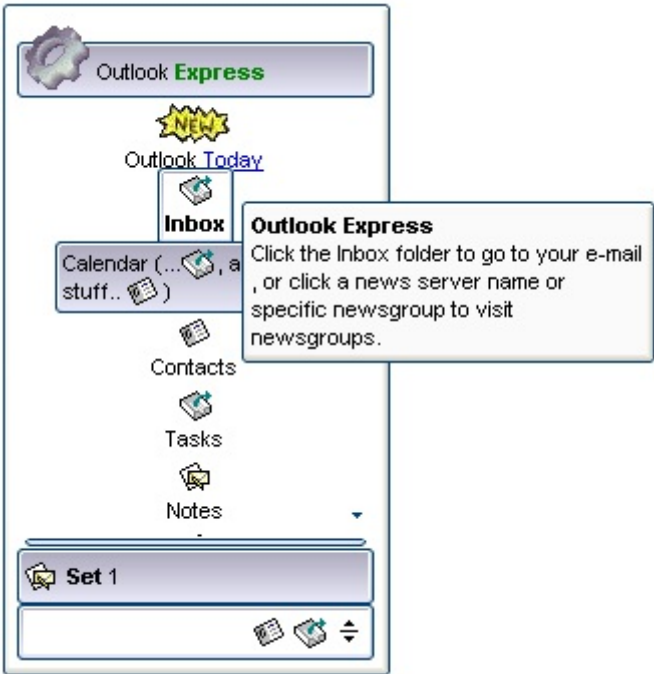
If the ToolTipDelay or ToolTipPopDelay property is 0, the control displays no tooltips. The [ToolTipDelay](#) property specifies the time in ms that passes before the ToolTip appears. Use the [ToolTipWidth](#) property to specify the width of the tooltip window. Use the [ToolTipFont](#) property to assign 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. Use the [ToolTip](#) property to specify the tooltip to be shown when the cursor hovers its caption. Use the [ShowToolTip](#) method to display a custom tooltip.

property ListBar.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. Use the [ToolTip](#) property to specify the tooltip to be shown when the cursor hovers its caption. Use the [ShowToolTip](#) method to display a custom tooltip.



property ListBar.UseVisualStyle as UIVisualThemeEnum

Specifies whether the control uses the current visual theme to display certain UI parts.

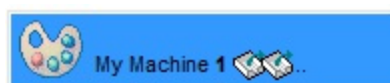
Type	Description
UIVisualThemeEnum	An UIVisualThemeEnum expression that specifies which UI parts of the control are shown using the current visual theme.


By default, the UseVisualStyle property is exDefaultVisualStyle, which means that all known UI parts are shown as in the current theme. The UseVisualStyle 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 UseVisualStyle property has effect only a current theme is selected for your desktop. The UseVisualStyle 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 UseVisualStyle property is exDefaultVisualStyle:



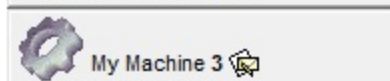
since the second screen shot shows the same data as the UseVisualStyle property is exNoVisualStyle:



 My Computer

 My Documents

 My Favorites



 Set 1



property ListBar.Version as String

Retrieves the control's version.

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

The version property specifies the control's version. For instance, the Version property for the DEMO UNICODE version of the control could be "1.0.1.5.DEMO.UNICODE".

property ListBar.VisualAppearance as Appearance

Retrieves the control's appearance.

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

Use the [Add](#) method to add or replace skins in 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.



ExListBar events

Tip The /COM object can be placed on a HTML page (with usage of the HTML object tag: <object classid="clsid:...">) using the class identifier: {41387A8B-6293-46CE-B9D8-55F49AE0DA60}. The object's program identifier is: "Exontrol.ListBar". The /COM object module is: "ExListBar.dll"

The Exontrol's ExListBar component supports the following events:

Name	Description
AddGroup	Occurs when a new group is added to collection.
AddItem	Occurs when a new item is added to a group.
AnchorClick	Occurs when an anchor element is clicked.
Click	Occurs when the user presses and then releases the left mouse button over the control.
DbClick	Occurs when the user dblclk the left mouse button over an object.
HighLightItem	Occurs when an item is highlighted.
KeyDown	Occurs when the user presses a key while an object has the focus.
KeyPress	Occurs when the user presses and releases an ANSI key.
KeyUp	Occurs when the user releases a key while an object has the focus.
MouseDown	Occurs when the user presses a mouse button.
MouseMove	Occurs when the user moves the mouse.
MouseUp	Occurs when the user releases a mouse button.
RClick	Fired when right mouse button is clicked
RemoveGroup	Fired when a group was removed.
RemoveItem	Fired when an item was removed.
SelectGroup	Occurs when a group is selected.
SelectItem	Occurs when an item is selected.
SelectShortcut	Fired when the user selects a new shortcut.

event AddGroup (Group as Group)

Occurs when a new group is added to collection.

Type	Description
Group as Group	A Group object that's added to the Groups collection.

Use the AddGroup event to notify your application that a new group was added to Groups collection. The [Add](#) method adds a new group to Groups collection. Use the [SelectedItem](#) property to get the index of the selected item. Use the [SelectGroup](#) property to get the index of the selected group. Use the [AddItem](#) method to add new items to the group. Use the [ItemHeight](#) property to specify the height for all items in the group. Use the [GroupHeight](#) property to specify the height for group captions. Use the [SmallIcons](#) property to specify the size of the icons being displayed.

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

```
C# private void AddGroup(object sender,exontrol.EXLISTBARLib.Group Group)
{
}
```

```
VB Private Sub AddGroup(ByVal sender As System.Object,ByVal Group As
exontrol.EXLISTBARLib.Group) Handles AddGroup
End Sub
```

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

```
C# private void AddGroup(object sender,
AxEXLISTBARLib._IListBarEvents_AddGroupEvent e)
{
}
```

```
C++ void OnAddGroup(LPDISPATCH Group)
{
}
```

```
C++ Builder void __fastcall AddGroup(TObject *Sender,Exlistbarlib_tlb::IGroup *Group)
{
}
```


Delphi

```
procedure AddGroup(ASender: TObject; Group : IGroup);  
begin  
end;
```

Delphi 8
(.NET
only)

```
procedure AddGroup(sender: System.Object; e:  
AxEXLISTBARLib._IListBarEvents_AddGroupEvent);  
begin  
end;
```

Power...

```
begin event AddGroup(oleobject Group)  
end event AddGroup
```

VB.NET

```
Private Sub AddGroup(ByVal sender As System.Object, ByVal e As  
AxEXLISTBARLib._IListBarEvents_AddGroupEvent) Handles AddGroup  
End Sub
```

VB6

```
Private Sub AddGroup(ByVal Group As EXLISTBARLibCtl.IGroup)  
End Sub
```

VBA

```
Private Sub AddGroup(ByVal Group As Object)  
End Sub
```

VFP

```
LPARAMETERS Group
```

Xbas...

```
PROCEDURE OnAddGroup(oListBar,Group)  
RETURN
```

Syntax for AddGroup event, **ICOM** version (others), on:

Java...

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

VBSc...

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

Visual
Data...

```
Procedure OnComAddGroup Variant IIGroup  
    Forward Send OnComAddGroup IIGroup  
End_Procedure
```

Visual
Objects

```
METHOD OCX_AddGroup(Group) CLASS MainDialog  
RETURN NIL
```

X++

```
void onEvent_AddGroup(COM _Group)  
{  
}
```

XBasic

```
function AddGroup as v (Group as OLE::Exontrol.ListBar.1::IGroup)  
end function
```

dBASE

```
function nativeObject_AddGroup(Group)  
return
```

The following VB sample changes the group's background color when adding a new group:

```
Private Sub ListBar1_AddGroup(ByVal Group As EXLISTBARLibCtl.IGroup)  
    With Group  
        .BackColor = RGB(0, 0, 255)  
        .Alignment = exRight  
        .ForeColor = vbWhite  
    End With  
End Sub
```

The following C++ sample changes the group's background color when adding a new group:

```
void OnAddGroupListbar1(LPDISPATCH Group)  
{  
    CGroup group( Group ); group.m_bAutoRelease = FALSE;  
    group.SetBackColor( RGB(0,0,255) );  
    group.SetAlignment( 2 /*exRight*/ );  
    group.SetForeColor( RGB(255,255,255) );  
}
```

The following VB.NET sample changes the group's background color when adding a new group:

```
Private Sub AxListBar1_AddGroup(ByVal sender As Object, ByVal e As
AxEXLISTBARLib._IListBarEvents_AddGroupEvent) Handles AxListBar1.AddGroup
    With e.group
        .BackColor = ToUInt32(Color.Blue)
        .ForeColor = ToUInt32(Color.White)
        .Alignment = EXLISTBARLib.AlignmentEnum.exRight
    End With
End Sub
```

where the ToUInt32 function converts a Color expression to OLE_COLOR expression,

```
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 group's background color when adding a new group:

```
private void axListBar1_AddGroup(object sender,
AxEXLISTBARLib._IListBarEvents_AddGroupEvent e)
{
    e.group.BackColor = ToUInt32(Color.Blue);
    e.group.ForeColor = ToUInt32(Color.White);
    e.group.Alignment = EXLISTBARLib.AlignmentEnum.exRight;
}
```

where the ToUInt32 function converts a Color expression to OLE_COLOR expression,

```
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 group's background color when adding a new group:

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

```
LPARAMETERS group
```

```
with group
```

```
  .BackColor = RGB(0,0,255)
```

```
  .ForeColor = RGB(255,255,255)
```

```
  .Alignment = 2 && exRight
```

```
endwith
```

event AddItem (Item as Item)

Occurs when a new item is added to a group.

Type	Description
Item as Item	An Item object that's added to the Group's items collection.

Use the AddItem event to notify your application that a new Item was added to Group. Use the [Group](#) property to find out the item's owner Group. The [AddItem](#) method fires the AddItem event each time when a new item was added to items Group collection. Use the [SelectItem](#) property to get the index of the selected item. Use the [SelectGroup](#) property to get the index of the selected group. Use the [Add](#) method to add new groups to the control.

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

C#	<pre>private void AddItem(object sender,exontrol.EXLISTBARLib.Item Item) { }</pre>
VB	<pre>Private Sub AddItem(ByVal sender As System.Object,ByVal Item As exontrol.EXLISTBARLib.Item) Handles AddItem End Sub</pre>

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

C#	<pre>private void AddItem(object sender, AxEXLISTBARLib._IListBarEvents_AddItemEvent e) { }</pre>
C++	<pre>void OnAddItem(LPDISPATCH Item) { }</pre>
C++ Builder	<pre>void __fastcall AddItem(TObject *Sender,Exlistbarlib_tlb::IItem *Item) { }</pre>
Delphi	<pre>procedure AddItem(ASender: TObject; Item : IItem);</pre>

```
begin  
end;
```

Delphi 8
(.NET
only)

```
procedure AddItem(sender: System.Object; e:  
AxEXLISTBARLib._IListBarEvents_AddItemEvent);  
begin  
end;
```

Powe...

```
begin event AddItem(oleobject Item)  
end event AddItem
```

VB.NET

```
Private Sub AddItem(ByVal sender As System.Object, ByVal e As  
AxEXLISTBARLib._IListBarEvents_AddItemEvent) Handles AddItem  
End Sub
```

VB6

```
Private Sub AddItem(ByVal Item As EXLISTBARLibCtl.IItem)  
End Sub
```

VBA

```
Private Sub AddItem(ByVal Item As Object)  
End Sub
```

VFP

```
LPARAMETERS Item
```

Xbas...

```
PROCEDURE OnAddItem(oListBar,Item)  
RETURN
```

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

Java...

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

VBSc...

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

Visual
Data...

```
Procedure OnComAddItem Variant lItem  
    Forward Send OnComAddItem lItem  
End_Procedure
```

Visual
Objects

```
METHOD OCX_AddItem(Item) CLASS MainDialog  
RETURN NIL
```

X++

```
void onEvent_AddItem(COM _Item)  
{  
}
```

XBasic

```
function AddItem as v (Item as OLE::Exontrol.ListBar.1::IItem)  
end function
```

dBASE

```
function nativeObject_AddItem(Item)  
return
```

The following VB sample changes the item's alignment when a new items is added to the first group:

```
Private Sub ListBar1_AddItem(ByVal Item As EXLISTBARLibCtl.IItem)  
    With Item  
        If (.Group.Index = 0) Then  
            .Alignment = exRight  
        End If  
    End With  
End Sub
```

The following C++ sample changes the item's alignment when a new items is added to the first group:

```
void OnAddItemListbar1(LPDISPATCH Item)  
{  
    CItem item( Item ); item.m_bAutoRelease = FALSE;  
    if ( item.GetGroup().GetIndex() == 0 )  
        item.SetAlignment( 2 /*exRight*/ );  
}
```

```
}
```

The following VB.NET sample changes the item's alignment when a new items is added to the first group:

```
Private Sub AxListBar1_AddItem(ByVal sender As Object, ByVal e As
AxEXLISTBARLib._IListBarEvents_AddItemEvent) Handles AxListBar1.AddItem
    With e.item
        If (.Group.Index = 0) Then
            .Alignment = EXLISTBARLib.AlignmentEnum.exRight
        End If
    End With
End Sub
```

The following C# sample changes the item's alignment when a new items is added to the first group:

```
private void axListBar1_AddItem(object sender,
AxEXLISTBARLib._IListBarEvents_AddItemEvent e)
{
    if (e.item.Group.Index == 0)
        e.item.Alignment = EXLISTBARLib.AlignmentEnum.exRight;
}
```

The following VFP sample changes the item's alignment when a new items is added to the first group:

```
*** ActiveX Control Event ***
LPARAMETERS item

with item
    If (.Group.Index = 0) Then
        .Alignment = 2 && exRight
    EndIf
endwith
```

```
Private Sub ListBar1_AddItem(ByVal Item As EXLISTBARLibCtl.IItem)
    If (Item.Group.Index = 0) Then
```



```
Item.Alignment = exRight
```

```
End If
```

```
End Sub
```

event **AnchorClick** (AnchorID as String, Options as String)

Occurs when an anchor element is clicked.

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

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

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

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

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

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

```
C# private void AnchorClick(object sender,
AxEXLISTBARLib._IListBarEvents_AnchorClickEvent e)
{
}
```

C++

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

**C++
Builder**

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

Delphi

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

**Delphi 8
(.NET
only)**

```
procedure AnchorClick(sender: System.Object; e:
AxEXLISTBARLib._IListBarEvents_AnchorClickEvent);
begin
end;
```

Powe...

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

VB.NET

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

VB6

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

VBA

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

VFP

```
LPARAMETERS AnchorID,Options
```

Xbas...

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

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

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

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

Visual
Data... Procedure OnComAnchorClick String IIAnchorID String IIOptions
Forward Send OnComAnchorClick IIAnchorID IIOptions
End_Procedure

Visual
Objects METHOD OCX_AnchorClick(AnchorID,Options) CLASS MainDialog
RETURN NIL

X++ void onEvent_AnchorClick(str _AnchorID,str _Options)
{
}

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

dBASE function nativeObject_AnchorClick(AnchorID,Options)
return

event Click ()

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

Type

Description

The Click event is fired when the user releases the left mouse button over the control. Use a [MouseDown](#) or [MouseUp](#) event procedure to specify actions that will occur when a mouse button is pressed or released. Unlike the Click and [DbClick](#) 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 [SelectedItem](#) property to get the index of the selected item. Use the [SelectGroup](#) property to get the index of the selected group.

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;
```

Power...

```
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(oListBar)  
RETURN
```

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

Java...

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

VBSc...

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

Visual
Data...

```
Procedure OnComClick  
Forward Send OnComClick
```

End_Procedure

Visual
Objects

METHOD OCX_Click() CLASS MainDialog
RETURN NIL

X++

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

XBasic

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

dBASE

```
function nativeObject_Click()
return
```

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

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

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

The DbtClick event is fired when the user dbl clicks on the control. Use the DbtClick event to notify your application that an item has been double-clicked. Use the [SelectItem](#) property to get the index of the selected item. Use the [SelectGroup](#) property to get the index of the selected group. Use the [ItemFromPoint](#) property to get the item over cursor. Use the [GroupFromPoint](#) property to get the group over cursor.

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

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

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

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

```
C# private void DbtClick(object sender, AxEXLISTBARLib._IListBarEvents_DbtClickEvent
e)
{
}
```

```
C++ void OnDbtClick(short Shift,long X,long Y)
{
}
```


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

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

Delphi 8 (.NET only) procedure DblClick(sender: System.Object; e: AxEXLISTBARLib._IListBarEvents_DblClickEvent);
begin
end;

PowerBuilder begin event DblClick(integer Shift,long X,long Y)
end event DblClick

VB.NET Private Sub DblClick(ByVal sender As System.Object, ByVal e As AxEXLISTBARLib._IListBarEvents_DblClickEvent) Handles DblClick
End Sub

VB6 Private Sub DblClick(Shift As Integer,X As Single,Y As Single)
End Sub

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

VFP LPARAMETERS Shift,X,Y

Xbase++ PROCEDURE OnDblClick(oListBar,Shift,X,Y)
RETURN

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

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

VBScript <SCRIPT LANGUAGE="VBScript">

```
Function DblClick(Shift,X,Y)
End Function
</SCRIPT>
```

Visual
Data...

```
Procedure OnComDblClick Short IIShift OLE_XPOS_PIXELS IIX OLE_YPOS_PIXELS  
IYY  
    Forward Send OnComDblClick IIShift IIX IYY  
End_Procedure
```

Visual
Objects

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

X++

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

XBasic

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

dBASE

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

The following VB sample displays the caption of the item being double clicked:

```
Private Sub ListBar1_DblClick(Shift As Integer, X As Single, Y As Single)  
    With ListBar1  
        Dim i As EXLISTBARLibCtl.Item  
        Set i = .ItemFromPoint(X / Screen.TwipsPerPixelX, Y / Screen.TwipsPerPixelY)  
        If (Not i Is Nothing) Then  
            Debug.Print i.Caption  
        End If  
    End With  
End Sub
```

The following C++ sample displays the caption of the item being double clicked:

```
void OnDblClickListbar1(short Shift, long X, long Y)
```

```

{
    CItem item = m_listbar.GetItemFromPoint( X, Y );
    if ( item.m_lpDispatch != NULL )
        OutputDebugString( item.GetCaption() );
}

```

The following VB.NET sample displays the caption of the item being double clicked:

```

Private Sub AxListBar1_DblClick(ByVal sender As Object, ByVal e As
AxEXLISTBARLib._IListBarEvents_DblClickEvent) Handles AxListBar1.DblClick
    With AxListBar1
        Dim i As EXLISTBARLib.Item = .get_ItemFromPoint(e.x, e.y)
        If (Not i Is Nothing) Then
            Debug.WriteLine(i.Caption)
        End If
    End With
End Sub

```

The following C# sample displays the caption of the item being double clicked:

```

private void axListBar1_DblClick(object sender,
AxEXLISTBARLib._IListBarEvents_DblClickEvent e)
{
    EXLISTBARLib.Item item = axListBar1.get_ItemFromPoint(e.x, e.y);
    if (item != null)
    {
        System.Diagnostics.Debug.WriteLine(item.Caption);
    }
}

```

The following VFP sample displays the caption of the item being double clicked:

```

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

With thisform.ListBar1
    local i
    i = .ItemFromPoint(x, y)
    If ( !isnull(i) ) Then

```

with i

wait window nowait .Caption

endwith

EndIf

EndWith

event HighlightItem (OldItem as Item,NewItem as Item)

Occurs when an item is highlighted.

Type	Description
OldItem as Item	An Item object that's un-highlighted.
NewItem as Item	An Item object that's highlighted.

Use the HighlightItem event to notify your application that an item is highlighted. The HighlightItem event occurs when cursor hovers the item. The [HighlightItemType](#) property specifies the way how the control marks the highlighted item. Use the [ForeColor](#) property to specify the item's foreground color.

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

C#private void HighlightItem(object sender,exontrol.EXLISTBARLib.Item OldItem,exontrol.EXLISTBARLib.Item NewItem){}

VBPrivate Sub HighlightItem(ByVal sender As System.Object,ByVal OldItem As exontrol.EXLISTBARLib.Item,ByVal NewItem As exontrol.EXLISTBARLib.Item)Handles HighlightItemEnd Sub

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

C#private void HighlightItem(object sender,AxEXLISTBARLib._IListBarEvents_HighLightItemEvent e){}

C++void OnHighlightItem(LPDISPATCH OldItem,LPDISPATCH NewItem){}

C++ Buildervoid __fastcall HighlightItem(TObject *Sender,Exlistbarlib_tlb::Item *OldItem,Exlistbarlib_tlb::Item *NewItem){}

Delphi procedure HighLightItem(ASender: TObject; OldItem : IItem;NewItem : IItem);
begin
end;

**Delphi 8
(.NET
only)** procedure HighLightItem(sender: System.Object; e:
AxEXLISTBARLib._IListBarEvents_HighLightItemEvent);
begin
end;

Powe... begin event HighLightItem(oleobject OldItem,oleobject NewItem)
end event HighLightItem

VB.NET Private Sub HighLightItem(ByVal sender As System.Object, ByVal e As
AxEXLISTBARLib._IListBarEvents_HighLightItemEvent) Handles HighLightItem
End Sub

VB6 Private Sub HighLightItem(ByVal OldItem As EXLISTBARLibCtl.IItem,ByVal NewItem
As EXLISTBARLibCtl.IItem)
End Sub

VBA Private Sub HighLightItem(ByVal OldItem As Object,ByVal NewItem As Object)
End Sub

VFP LPARAMETERS OldItem,NewItem

Xbas... PROCEDURE OnHighLightItem(oListBar,OldItem,NewItem)
RETURN

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

Java... <SCRIPT EVENT="HighLightItem(OldItem,NewItem)" LANGUAGE="JScript">
</SCRIPT>

VBSc... <SCRIPT LANGUAGE="VBScript">
Function HighLightItem(OldItem,NewItem)
End Function
</SCRIPT>

Visual Data... Procedure OnComHighLightItem Variant IIOldItem Variant IINewItem
Forward Send OnComHighLightItem IIOldItem IINewItem
End_Procedure

Visual Objects METHOD OCX_HighLightItem(OldItem,NewItem) CLASS MainDialog
RETURN NIL

X++ void onEvent_HighLightItem(COM _OldItem,COM _NewItem)
{
}

XBasic function HighLightItem as v (OldItem as OLE::Exontrol.ListBar.1::Item,NewItem as
OLE::Exontrol.ListBar.1::Item)
end function

dBASE function nativeObject_HighLightItem(OldItem,NewItem)
return

The following VB sample bolds the highlighted item:

```
Private Sub ListBar1_HighLightItem(ByVal OldItem As EXLISTBARLibCtl.Item, ByVal  
NewItem As EXLISTBARLibCtl.Item)  
    If Not OldItem Is Nothing Then  
        OldItem.Bold = False  
    End If  
    If Not NewItem Is Nothing Then  
        NewItem.Bold = True  
    End If  
End Sub
```

The following C++ sample bolds the highlighted item:

```
void OnHighLightItemListbar1(LPDISPATCH OldItem, LPDISPATCH NewItem)  
{  
    CItem oldItem( OldItem ); oldItem.m_bAutoRelease = FALSE;  
    CItem newItem( NewItem ); newItem.m_bAutoRelease = FALSE;  
    if ( oldItem.m_lpDispatch != NULL )  
        oldItem.SetBold( FALSE );  
}
```

```
if ( newItem.m_lpDispatch != NULL )
    newItem.SetBold( TRUE );
}
```

The following VB.NET sample bolds the highlighted item:

```
Private Sub AxListBar1_HighLightItem(ByVal sender As Object, ByVal e As
AxEXLISTBARLib._IListBarEvents_HighLightItemEvent) Handles AxListBar1.HighLightItem
    If Not e.oldItem Is Nothing Then
        e.oldItem.Bold = False
    End If
    If Not e.newItem Is Nothing Then
        e.newItem.Bold = True
    End If
End Sub
```

The following C# sample bolds the highlighted item:

```
private void axListBar1_HighLightItem(object sender,
AxEXLISTBARLib._IListBarEvents_HighLightItemEvent e)
{
    if (e.oldItem != null)
        e.oldItem.Bold = false;
    if (e.newItem != null)
        e.newItem.Bold = true;
}
```

The following VFP sample bolds the highlighted item:

```
*** ActiveX Control Event ***
LPARAMETERS olditem, newitem

If !isnull(OldItem)
    OldItem.Bold = .f.
EndIf
If !isnull(NewItem)
    NewItem.Bold = .t.
EndIf
```


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

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

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

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

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

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

Use the [SelectedItem](#) property to get the index of the selected item. Use the [SelectGroup](#) property to get the index of the selected group.

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,
AxEXLISTBARLib._IListBarEvents_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:
AxEXLISTBARLib._IListBarEvents_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
AxEXLISTBARLib._IListBarEvents_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(oListBar,KeyCode,Shift)
RETURN
```

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

Java...

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

VBSc...

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

Visual
Data...

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

Visual
Objects

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

X++

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

XBasic

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

dBASE

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

event KeyPress (ByRef KeyAscii as Integer)

Occurs when the user presses and releases an ANSI key.

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

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

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,
AxEXLISTBARLib._IListBarEvents_KeyPressEvent e)
{
}
```

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

```
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:
AxEXLISTBARLib._IListBarEvents_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
AxEXLISTBARLib._IListBarEvents_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(oListBar,KeyAscii)
RETURN
```

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

Java...

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

VBS...

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

**Visual
Data...**

```
Procedure OnComKeyPress Short IIKeyAscii  
    Forward Send OnComKeyPress IIKeyAscii  
End_Procedure
```

**Visual
Objects**

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

X++

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

XBasic

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

dBASE

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

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

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

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

Use the KeyUp event procedure to respond to the releasing of a key. Use the [SelectedItem](#) property to get the index of the selected item. Use the [SelectGroup](#) property to get the index of the selected group.

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,
AxEXLISTBARLib._IListBarEvents_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: AxEXLISTBARLib._IListBarEvents_KeyUpEvent);
begin
end;

PowerBuilder begin event KeyUp(integer KeyCode,integer Shift)
end event KeyUp

VB.NET Private Sub KeyUpEvent(ByVal sender As System.Object, ByVal e As AxEXLISTBARLib._IListBarEvents_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

Xbase++ PROCEDURE OnKeyUp(oListBar,KeyCode,Shift)
RETURN

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

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

VBScript <SCRIPT LANGUAGE="VBScript">

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

Visual
Data...

```
Procedure OnComKeyUp Short Integer KeyCode Short Integer Shift
    Forward Send OnComKeyUp Integer KeyCode Integer Shift
End Procedure
```

Visual
Objects

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

C++

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

XBasic

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

dBASE

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

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

Occurs when the user presses a mouse button.

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

Use a MouseDown or [MouseUp](#) event procedure to specify actions that will occur when a mouse button is pressed or released. Unlike the [Click](#) and [DbClick](#) 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 [SelectedItem](#) property to get the index of the selected item. Use the [SelectGroup](#) property to get the index of the selected group. Use the [ItemFromPoint](#) property to get the item over cursor. Use the [GroupFromPoint](#) property to get the group over cursor.

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,
```

```
AxEXLISTBARLib._IListBarEvents_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: AxEXLISTBARLib._IListBarEvents_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 AxEXLISTBARLib._IListBarEvents_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(oListBar,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 IButton Short IShift OLE_XPOS_PIXELS IIX  
OLE_YPOS_PIXELS IY  
Forward Send OnComMouseDown IButton IShift IIX IY  
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.ListBar.1::OLE_XPOS_PIXELS,Y as  
OLE::Exontrol.ListBar.1::OLE_YPOS_PIXELS)  
end function
```

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

The following VB sample displays the caption of the group being clicked:

```
Private Sub ListBar1_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As  
Single)
```

```

With ListBar1
    Dim g As Group
    Set g = .GroupFromPoint(X / Screen.TwipsPerPixelX, Y / Screen.TwipsPerPixelY)
    If Not g Is Nothing Then
        MsgBox g.Caption
    End If
End With
End Sub

```

The following VB sample displays the caption of the item being clicked:

```

Private Sub ListBar1_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
    With ListBar1
        Dim i As Item
        Set i = .ItemFromPoint(X / Screen.TwipsPerPixelX, Y / Screen.TwipsPerPixelY)
        If Not i Is Nothing Then
            MsgBox i.Caption
        End If
    End With
End Sub

```

The following C++ sample displays the caption of the group being clicked:

```

void OnMouseDownListbar1(short Button, short Shift, long X, long Y)
{
    CGroup group = m_listbar.GetGroupFromPoint( X, Y );
    if ( group.m_lpDispatch != NULL )
        MessageBox( group.GetCaption() );
}

```

The following C++ sample displays the caption of the item being clicked:

```

void OnMouseDownListbar1(short Button, short Shift, long X, long Y)
{
    CItem item = m_listbar.GetItemFromPoint( X, Y );
    if ( item.m_lpDispatch != NULL )
        MessageBox( item.GetCaption() );
}

```

The following VB.NET sample displays the caption of the group being clicked:

```
Private Sub AxListBar1_MouseDownEvent(ByVal sender As Object, ByVal e As
AxEXLISTBARLib._IListBarEvents_MouseDownEvent) Handles AxListBar1.MouseDownEvent
    With AxListBar1
        Dim g As EXLISTBARLib.Group = .get_GroupFromPoint(e.x, e.y)
        If Not g Is Nothing Then
            MsgBox(g.Caption)
        End If
    End With
End Sub
```

The following VB.NET sample displays the caption of the item being clicked:

```
Private Sub AxListBar1_MouseDownEvent(ByVal sender As Object, ByVal e As
AxEXLISTBARLib._IListBarEvents_MouseDownEvent) Handles AxListBar1.MouseDownEvent
    With AxListBar1
        Dim i As EXLISTBARLib.Item = .get_ItemFromPoint(e.x, e.y)
        If Not i Is Nothing Then
            MsgBox(i.Caption)
        End If
    End With
End Sub
```

The following C# sample displays the caption of the group being clicked:

```
private void axListBar1_MouseDownEvent(object sender,
AxEXLISTBARLib._IListBarEvents_MouseDownEvent e)
{
    EXLISTBARLib.Group g = axListBar1.get_GroupFromPoint(e.x, e.y);
    if (g != null)
        MessageBox.Show(g.Caption);
}
```

The following C# sample displays the caption of the item being clicked:

```
private void axListBar1_MouseDownEvent(object sender,
AxEXLISTBARLib._IListBarEvents_MouseDownEvent e)
```

```

{
    EXLISTBARLib.Item i = axListBar1.get_ItemFromPoint(e.x, e.y);
    if (i != null)
        MessageBox.Show(i.Caption);
}

```

The following VFP sample displays the caption of the group being clicked:

```

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

With thisform.ListBar1
    local g
    g = .GroupFromPoint(x , y)
    If !isnull(g)
        with g
            wait window nowait .Caption
        endwith
    EndIf
EndWith

```

The following VFP sample displays the caption of the item being clicked:

```

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

With thisform.ListBar1
    local i
    i = .ItemFromPoint(x , y)
    If !isnull(i)
        with i
            wait window nowait .Caption
        endwith
    EndIf
EndWith

```


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

Occurs when the user moves the mouse.

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

The MouseEventArgs event is generated continually as the mouse pointer moves across objects. Unless another object has captured the mouse, an object recognizes a MouseEventArgs event whenever the mouse position is within its borders. Use the [ItemFromPoint](#) property to get the item over cursor. Use the [GroupFromPoint](#) property to get the group over cursor.

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

C#

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

VB

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

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

C#

```
private void MouseEventArgsEvent(object sender,
AxEXLISTBARLib._IListBarEvents_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:
AxEXLISTBARLib._IListBarEvents_MouseMoveEvent);
begin
end;
```

Power...

```
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
AxEXLISTBARLib._IListBarEvents_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(oListBar,Button,Shift,X,Y)
RETURN
```

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

Java...	<pre><SCRIPT EVENT="MouseMove(Button,Shift,X,Y)" LANGUAGE="JScript"> </SCRIPT></pre>
VBSc...	<pre><SCRIPT LANGUAGE="VBScript"> Function MouseMove(Button,Shift,X,Y) End Function </SCRIPT></pre>
Visual Data...	<pre>Procedure OnComMouseMove Short IButton Short IShift OLE_XPOS_PIXELS IIX OLE_YPOS_PIXELS IY Forward Send OnComMouseMove IButton IShift IIX IY End_Procedure</pre>
Visual Objects	<pre>METHOD OCX_MouseMove(Button,Shift,X,Y) CLASS MainDialog RETURN NIL</pre>
X++	<pre>void onEvent_MouseMove(int _Button,int _Shift,int _X,int _Y) { }</pre>
XBasic	<pre>function MouseMove as v (Button as N,Shift as N,X as OLE::Exontrol.ListBar.1::OLE_XPOS_PIXELS,Y as OLE::Exontrol.ListBar.1::OLE_YPOS_PIXELS) end function</pre>
dBASE	<pre>function nativeObject_MouseMove(Button,Shift,X,Y) return</pre>

The following VB sample displays the caption of the group from the cursor:

```
Private Sub ListBar1_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
    With ListBar1
        Dim g As Group
        Set g = .GroupFromPoint(X / Screen.TwipsPerPixelX, Y / Screen.TwipsPerPixelY)
```

```
If Not g Is Nothing Then
    Debug.Print g.Caption
End If
End With
End Sub
```

The following VB sample displays the caption of the item from the cursor:

```
Private Sub ListBar1_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
    With ListBar1
        Dim i As Item
        Set i = .ItemFromPoint(X / Screen.TwipsPerPixelX, Y / Screen.TwipsPerPixelY)
        If Not i Is Nothing Then
            Debug.Print i.Caption
        End If
    End With
End Sub
```

The following C++ sample displays the caption of the group from the cursor:

```
void OnMouseMoveListbar1(short Button, short Shift, long X, long Y)
{
    CGroup group = m_listbar.GetGroupFromPoint( X, Y );
    if ( group.m_lpDispatch != NULL )
        OutputDebugString( group.GetCaption() );
}
```

The following C++ sample displays the caption of the item from the cursor:

```
void OnMouseMoveListbar1(short Button, short Shift, long X, long Y)
{
    CItem item = m_listbar.GetItemFromPoint( X, Y );
    if ( item.m_lpDispatch != NULL )
        OutputDebugString( item.GetCaption() );
}
```

The following VB.NET sample displays the caption of the group from the cursor:

```

Private Sub AxListBar1_MouseMoveEvent(ByVal sender As Object, ByVal e As
AxEXLISTBARLib._IListBarEvents_MouseMoveEvent) Handles AxListBar1.MouseDownEvent
    With AxListBar1
        Dim g As EXLISTBARLib.Group = .get_GroupFromPoint(e.x, e.y)
        If Not g Is Nothing Then
            Debug.WriteLine(g.Caption)
        End If
    End With
End Sub

```

The following VB.NET sample displays the caption of the item from the cursor:

```

Private Sub AxListBar1_MouseMoveEvent(ByVal sender As Object, ByVal e As
AxEXLISTBARLib._IListBarEvents_MouseMoveEvent) Handles AxListBar1.MouseDownEvent
    With AxListBar1
        Dim i As EXLISTBARLib.Item = .get_ItemFromPoint(e.x, e.y)
        If Not i Is Nothing Then
            Debug.WriteLine(i.Caption)
        End If
    End With
End Sub

```

The following C# sample displays the caption of the group from the cursor:

```

private void axListBar1_MouseMoveEvent(object sender,
AxEXLISTBARLib._IListBarEvents_MouseMoveEvent e)
{
    EXLISTBARLib.Group g = axListBar1.get_GroupFromPoint(e.x, e.y);
    if (g != null)
        System.Diagnostics.Debug.WriteLine(g.Caption);
}

```

The following C# sample displays the caption of the item from the cursor:

```

private void axListBar1_MouseMoveEvent(object sender,
AxEXLISTBARLib._IListBarEvents_MouseMoveEvent e)
{
    EXLISTBARLib.Item i = axListBar1.get_ItemFromPoint(e.x, e.y);
}

```

```
if (i != null)
    System.Diagnostics.Debug.WriteLine(i.Caption);
}
```

The following VFP sample displays the caption of the group from the cursor:

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

With thisform.ListBar1
    local g
    g = .GroupFromPoint(x , y)
    If !isNull(g)
        with g
            wait window nowait .Caption
        endwhile
    EndIf
EndWith
```

The following VFP sample displays the caption of the item from the cursor:

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

With thisform.ListBar1
    local i
    i = .ItemFromPoint(x , y)
    If !isNull(i)
        with i
            wait window nowait .Caption
        endwhile
    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 [DbClick](#) 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 [ItemFromPoint](#) property to get the item over cursor. Use the [GroupFromPoint](#) property to get the group over cursor.

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,
AxEXLISTBARLib._IListBarEvents_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:
AxEXLISTBARLib._IListBarEvents_MouseUpEvent);
begin
end;
```

Power...

```
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
AxEXLISTBARLib._IListBarEvents_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(oListBar,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 IButton Short IShift OLE_XPOS_PIXELS IIX
OLE_YPOS_PIXELS IY
 Forward Send OnComMouseUp IButton IShift IIX IY
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.ListBar.1::OLE_XPOS_PIXELS,Y as
OLE::Exontrol.ListBar.1::OLE_YPOS_PIXELS)
end function

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

The following VB sample displays the caption of the group being clicked:

```
Private Sub ListBar1_MouseUp(Button As Integer, Shift As Integer, X As Single, Y As Single)
    With ListBar1
        Dim g As Group
```

```

Set g = .GroupFromPoint(X / Screen.TwipsPerPixelX, Y / Screen.TwipsPerPixelY)
If Not g Is Nothing Then
    MsgBox g.Caption
End If
End With
End Sub

```

The following VB sample displays the caption of the item being clicked:

```

Private Sub ListBar1_MouseUp(Button As Integer, Shift As Integer, X As Single, Y As Single)
    With ListBar1
        Dim i As Item
        Set i = .ItemFromPoint(X / Screen.TwipsPerPixelX, Y / Screen.TwipsPerPixelY)
        If Not i Is Nothing Then
            MsgBox i.Caption
        End If
    End With
End Sub

```

The following C++ sample displays the caption of the group being clicked:

```

void OnMouseUpListbar1(short Button, short Shift, long X, long Y)
{
    CGroup group = m_listbar.GetGroupFromPoint( X, Y );
    if ( group.m_lpDispatch != NULL )
        MessageBox( group.GetCaption() );
}

```

The following C++ sample displays the caption of the item being clicked:

```

void OnMouseUpListbar1(short Button, short Shift, long X, long Y)
{
    CItem item = m_listbar.GetItemFromPoint( X, Y );
    if ( item.m_lpDispatch != NULL )
        MessageBox( item.GetCaption() );
}

```

The following VB.NET sample displays the caption of the group being clicked:

```

Private Sub AxListBar1_MouseUpEvent(ByVal sender As Object, ByVal e As
AxEXLISTBARLib._IListBarEvents_MouseUpEvent) Handles AxListBar1.MouseDownEvent
    With AxListBar1
        Dim g As EXLISTBARLib.Group = .get_GroupFromPoint(e.x, e.y)
        If Not g Is Nothing Then
            MsgBox(g.Caption)
        End If
    End With
End Sub

```

The following VB.NET sample displays the caption of the item being clicked:

```

Private Sub AxListBar1_MouseUpEvent(ByVal sender As Object, ByVal e As
AxEXLISTBARLib._IListBarEvents_MouseUpEvent) Handles AxListBar1.MouseDownEvent
    With AxListBar1
        Dim i As EXLISTBARLib.Item = .get_ItemFromPoint(e.x, e.y)
        If Not i Is Nothing Then
            MsgBox(i.Caption)
        End If
    End With
End Sub

```

The following C# sample displays the caption of the group being clicked:

```

private void axListBar1_MouseUpEvent(object sender,
AxEXLISTBARLib._IListBarEvents_MouseUpEvent e)
{
    EXLISTBARLib.Group g = axListBar1.get_GroupFromPoint(e.x, e.y);
    if (g != null)
        MessageBox.Show(g.Caption);
}

```

The following C# sample displays the caption of the item being clicked:

```

private void axListBar1_MouseUpEvent(object sender,
AxEXLISTBARLib._IListBarEvents_MouseUpEvent e)
{
    EXLISTBARLib.Item i = axListBar1.get_ItemFromPoint(e.x, e.y);
}

```

```
if (i != null)
    MessageBox.Show(i.Caption);
}
```

The following VFP sample displays the caption of the group being clicked:

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

With thisform.ListBar1
    local g
    g = .GroupFromPoint(x , y)
    If !isNull(g)
        with g
            wait window nowait .Caption
        endwhile
    EndIf
EndWith
```

The following VFP sample displays the caption of the item being clicked:

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

With thisform.ListBar1
    local i
    i = .ItemFromPoint(x , y)
    If !isNull(i)
        with i
            wait window nowait .Caption
        endwhile
    EndIf
EndWith
```

event RClick ()

Fired when right mouse button is clicked

Type	Description
------	-------------

The RClick event is fired each time the user releases the right mouse button over the control. Use the [MouseDown](#) or [MouseUp](#) event if you need the cursor coordinates. Else, you can use the GetCursorPos API function. Use the [ItemFromPoint](#) property to get the item over cursor. Use the [GroupFromPoint](#) property to get the group over cursor.

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

C#	<pre>private void RClick(object sender) { }</pre>
VB	<pre>Private Sub RClick(ByVal sender As System.Object) Handles RClick End Sub</pre>

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

C#	<pre>private void RClick(object sender, EventArgs e) { }</pre>
C++	<pre>void OnRClick() { }</pre>
C++ Builder	<pre>void __fastcall RClick(TObject *Sender) { }</pre>
Delphi	<pre>procedure RClick(ASender: TObject;); begin end;</pre>
Delphi 8 (.NET only)	<pre>procedure RClick(sender: System.Object; e: System.EventArgs); begin end;</pre>

Powe... begin event RClick()
end event RClick

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

VB6 Private Sub RClick()
End Sub

VBA Private Sub RClick()
End Sub

VFP LPARAMETERS nop

Xbas... PROCEDURE OnRClick(oListBar)
RETURN

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

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

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

Visual
Data... Procedure OnComRClick
Forward Send OnComRClick
End_Procedure

Visual
Objects METHOD OCX_RClick() CLASS MainDialog
RETURN NIL

X++ void onEvent_RClick()

```
{  
}
```

XBasic

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

dBASE

```
function nativeObject_RClick()  
return
```


event RemoveGroup (Group as Group)

Fired when a group was removed.

Type	Description
Group as Group	A Group object being removed.

Use the RemoveGroup event to notify your application that a group was released. Use the RemoveGroup event to release any extra data stored by the group. The [Remove](#) method fires the RemoveGroup event for each group removed. Use the RemoveItem event to notify your application that an item was deleted. Use the [UserData](#) property to associate an extra data to a group.

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

C#	<pre>private void RemoveGroup(object sender,exontrol.EXLISTBARLib.Group Group) { }</pre>
VB	<pre>Private Sub RemoveGroup(ByVal sender As System.Object,ByVal Group As exontrol.EXLISTBARLib.Group) Handles RemoveGroup End Sub</pre>

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

C#	<pre>private void RemoveGroup(object sender, AxEXLISTBARLib._IListBarEvents_RemoveGroupEvent e) { }</pre>
C++	<pre>void OnRemoveGroup(LPDISPATCH Group) { }</pre>
C++ Builder	<pre>void __fastcall RemoveGroup(TObject *Sender,Exlistbarlib_tlb::IGroup *Group) { }</pre>
Delphi	<pre>procedure RemoveGroup(ASender: TObject; Group : IGroup); begin</pre>

```
end;
```

Delphi 8
(.NET
only)

```
procedure RemoveGroup(sender: System.Object; e:  
AxEXLISTBARLib._IListBarEvents_RemoveGroupEvent);  
begin  
end;
```

Powe...

```
begin event RemoveGroup(oleobject Group)  
end event RemoveGroup
```

VB.NET

```
Private Sub RemoveGroup(ByVal sender As System.Object, ByVal e As  
AxEXLISTBARLib._IListBarEvents_RemoveGroupEvent) Handles RemoveGroup  
End Sub
```

VB6

```
Private Sub RemoveGroup(ByVal Group As EXLISTBARLibCtl.IGroup)  
End Sub
```

VBA

```
Private Sub RemoveGroup(ByVal Group As Object)  
End Sub
```

VFP

```
LPARAMETERS Group
```

Xbas...

```
PROCEDURE OnRemoveGroup(oListBar,Group)  
RETURN
```

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

Java...

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

VBSc...

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

Visual
Data...

```
Procedure OnComRemoveGroup Variant llGroup  
Forward Send OnComRemoveGroup llGroup
```

End_Procedure

Visual
Objects

METHOD OCX_RemoveGroup(Group) CLASS MainDialog
RETURN NIL

X++

```
void onEvent_RemoveGroup(COM _Group)
{
}
```

XBasic

```
function RemoveGroup as v (Group as OLE::Exontrol.ListBar.1::IGroup)
end function
```

dBASE

```
function nativeObject_RemoveGroup(Group)
return
```

The following VB sample displays the caption of the group being removed:

```
Private Sub ListBar1_RemoveGroup(ByVal Group As EXLISTBARLibCtl.IGroup)
    Debug.Print Group.Caption
End Sub
```

The following C++ sample displays the caption of the group being removed:

```
void OnRemoveGroupListbar1(LPDISPATCH Group)
{
    CGroup group( Group ); group.m_bAutoRelease = FALSE;
    CString strOutput;
    strOutput.Format("%s", (LPCTSTR)group.GetCaption() );
    OutputDebugString( strOutput );
}
```

The following VB.NET sample displays the caption of the group being removed:

```
Private Sub AxListBar1_RemoveGroup(ByVal sender As Object, ByVal e As
AxEXLISTBARLib._IListBarEvents_RemoveGroupEvent) Handles AxListBar1.RemoveGroup
    Debug.WriteLine(e.group.Caption)
End Sub
```

The following C# sample displays the caption of the group being removed:

```
private void axListBar1_RemoveGroup(object sender,  
AxEXLISTBARLib._IListBarEvents_RemoveGroupEvent e)  
{  
    System.Diagnostics.Debug.WriteLine(e.group.Caption);  
}
```

The following VFP sample displays the caption of the group being removed:

```
*** ActiveX Control Event ***  
LPARAMETERS group  
  
with group  
    wait window nowait .Caption  
endwith
```

event RemoveItem (Item as Item)

Fired when an item was removed.

Type	Description
Item as Item	An Item object that's removed.

Use the RemoveItem event to notify your application that an Item is removed. Use the RemoveItem event to release any extra data hold by an Item object. The [RemoveItem](#) method fires the RemoveItem event each time when an item is removed. The control fires the [RemoveGroup](#) event when a group is removed. Use the [UserData](#) property to associate an extra data to an item. Use the [Visible](#) property to hide an item.

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

C#

```
private void RemoveItem(object sender,exontrol.EXLISTBARLib.Item Item)
{
}
```

VB

```
Private Sub RemoveItem(ByVal sender As System.Object,ByVal Item As
exontrol.EXLISTBARLib.Item) Handles RemoveItem
End Sub
```

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

C#

```
private void RemoveItem(object sender,
AxEXLISTBARLib._IListBarEvents_RemoveItemEvent e)
{
}
```

C++

```
void OnRemoveItem(LPDISPATCH Item)
{
}
```

C++ Builder

```
void __fastcall RemoveItem(TObject *Sender,Exlistbarlib_tlb::IItem *Item)
{
}
```

Delphi

```
procedure RemoveItem(ASender: TObject; Item : IItem);
begin
```

```
end;
```

Delphi 8
(.NET
only)

```
procedure RemoveItem(sender: System.Object; e:  
AxEXLISTBARLib._IListBarEvents_RemoveItemEvent);  
begin  
end;
```

Powe...

```
begin event RemoveItem(oleobject Item)  
end event RemoveItem
```

VB.NET

```
Private Sub RemoveItem(ByVal sender As System.Object, ByVal e As  
AxEXLISTBARLib._IListBarEvents_RemoveItemEvent) Handles RemoveItem  
End Sub
```

VB6

```
Private Sub RemoveItem(ByVal Item As EXLISTBARLibCtl.IItem)  
End Sub
```

VBA

```
Private Sub RemoveItem(ByVal Item As Object)  
End Sub
```

VFP

```
LPARAMETERS Item
```

Xbas...

```
PROCEDURE OnRemoveItem(oListBar,Item)  
RETURN
```

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

Java...

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

VBSc...

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

Visual
Data...

```
Procedure OnComRemoveItem Variant IItem  
Forward Send OnComRemoveItem IItem
```

End_Procedure

Visual
Objects

METHOD OCX_RemoveItem(Item) CLASS MainDialog
RETURN NIL

X++

```
void onEvent_RemoveItem(COM _Item)
{
}
```

XBasic

```
function RemoveItem as v (Item as OLE::Exontrol.ListBar.1::IItem)
end function
```

dBASE

```
function nativeObject_RemoveItem(Item)
return
```

The following VB sample prints the caption of the item being removed:

```
Private Sub ListBar1_RemoveItem(ByVal Item As EXLISTBARLibCtl.IItem)
    With Item
        Debug.Print .Caption
    End With
End Sub
```

The following C++ sample prints the caption of the item being removed:

```
void OnRemoveItemListbar1(LPDISPATCH Item)
{
    CItem item( Item ); item.m_bAutoRelease = FALSE;
    CString strOutput;
    strOutput.Format("%s", (LPCTSTR)item.GetCaption() );
    OutputDebugString( strOutput );
}
```

The following VB.NET sample prints the caption of the item being removed:

```
Private Sub AxListBar1_RemoveItem(ByVal sender As Object, ByVal e As
AxEXLISTBARLib._IListBarEvents_RemoveItemEvent) Handles AxListBar1.RemoveItem
    With e.item
```

```
        Debug.WriteLine(.Caption)
    End With
End Sub
```

The following C# sample prints the caption of the item being removed:

```
private void axListBar1_RemoveItem(object sender,
AxEXLISTBARLib._IListBarEvents_RemoveItemEvent e)
{
    System.Diagnostics.Debug.WriteLine(e.item.Caption);
}
```

The following VFP sample prints the caption of the item being removed:

```
*** ActiveX Control Event ***
LPARAMETERS item

with item
    wait window nowait .Caption
endwith
```


event **SelectGroup** (OldGroup as Group, NewGroup as Group)

Occurs when a group is selected.

Type	Description
OldGroup as Group	A Group object that's unselected.
NewGroup as Group	A Group object that's selected.

Use the **SelectGroup** event to notify your application that a new group was selected. Use the [SelectGroup](#) property to get the index of the selected group. Use the [SelectItem](#) event to notify your application that a new item was selected. Use the [SelectItem](#) property to retrieve the index of selected in the group. Use the [Caption](#) property to get the caption of the item. Use the [Caption](#) property to get the caption of the group.

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

C#

```
private void SelectGroup(object sender,exontrol.EXLISTBARLib.Group OldGroup,exontrol.EXLISTBARLib.Group NewGroup)
{
}
```

VB

```
Private Sub SelectGroup(ByVal sender As System.Object,ByVal OldGroup As exontrol.EXLISTBARLib.Group,ByVal NewGroup As exontrol.EXLISTBARLib.Group)
Handles SelectGroup
End Sub
```

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

C#

```
private void SelectGroup(object sender,
AxEXLISTBARLib._IListBarEvents_SelectGroupEvent e)
{
}
```

C++

```
void OnSelectGroup(LPDISPATCH OldGroup,LPDISPATCH NewGroup)
{
}
```

C++ Builder

```
void __fastcall SelectGroup(TObject *Sender,Exlistbarlib_tlb::IGroup *OldGroup,Exlistbarlib_tlb::IGroup *NewGroup)
{
}
```

```
}
```

```
Delphi procedure SelectGroup(ASender: TObject; OldGroup : IGroup;NewGroup :  
IGroup);  
begin  
end;
```

```
Delphi 8 (.NET only) procedure SelectGroup(sender: System.Object; e:  
AxEXLISTBARLib._IListBarEvents_SelectGroupEvent);  
begin  
end;
```

```
Powe... begin event SelectGroup(oleobject OldGroup,oleobject NewGroup)  
end event SelectGroup
```

```
VB.NET Private Sub SelectGroup(ByVal sender As System.Object, ByVal e As  
AxEXLISTBARLib._IListBarEvents_SelectGroupEvent) Handles SelectGroup  
End Sub
```

```
VB6 Private Sub SelectGroup(ByVal OldGroup As EXLISTBARLibCtl.IGroup,ByVal  
NewGroup As EXLISTBARLibCtl.IGroup)  
End Sub
```

```
VBA Private Sub SelectGroup(ByVal OldGroup As Object,ByVal NewGroup As Object)  
End Sub
```

```
VFP LPARAMETERS OldGroup,NewGroup
```

```
Xbas... PROCEDURE OnSelectGroup(oListBar,OldGroup,NewGroup)  
RETURN
```

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

```
Java... <SCRIPT EVENT="SelectGroup(OldGroup,NewGroup)" LANGUAGE="JScript">  
</SCRIPT>
```

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

```
Function SelectGroup(OldGroup,NewGroup)
End Function
</SCRIPT>
```

Visual Data...

```
Procedure OnComSelectGroup Variant IIOldGroup Variant IINewGroup
    Forward Send OnComSelectGroup IIOldGroup IINewGroup
End_Procedure
```

Visual Objects

```
METHOD OCX_SelectGroup(OldGroup,NewGroup) CLASS MainDialog
RETURN NIL
```

X++

```
void onEvent_SelectGroup(COM _OldGroup,COM _NewGroup)
{
}
```

XBasic

```
function SelectGroup as v (OldGroup as OLE::Exontrol.ListBar.1::IGroup,NewGroup
as OLE::Exontrol.ListBar.1::IGroup)
end function
```

dBASE

```
function nativeObject_SelectGroup(OldGroup,NewGroup)
return
```

The following VB sample displays the caption of the group being selected:

```
Private Sub ListBar1_SelectGroup(ByVal Group As EXLISTBARLibCtl.IGroup)
    Debug.Print Group.Caption
End Sub
```

The following C++ sample displays the caption of the group being selected:

```
void OnSelectGroupListbar1(LPDISPATCH Group)
{
    CGroup group( Group ); group.m_bAutoRelease = FALSE;
    CString strOutput;
    strOutput.Format("%s", (LPCTSTR)group.GetCaption() );
    OutputDebugString( strOutput );
}
```

The following VB.NET sample displays the caption of the group being selected:

```
Private Sub AxListBar1_SelectGroup(ByVal sender As Object, ByVal e As  
AxEXLISTBARLib._IListBarEvents_SelectGroupEvent) Handles AxListBar1.SelectGroup  
    With e.group  
        Debug.WriteLine(.Caption)  
    End With  
End Sub
```

The following C# sample displays the caption of the group being selected:

```
private void axListBar1_SelectGroup(object sender,  
AxEXLISTBARLib._IListBarEvents_SelectGroupEvent e)  
{  
    System.Diagnostics.Debug.WriteLine(e.group.Caption);  
}
```

The following VFP sample displays the caption of the group being selected:

```
*** ActiveX Control Event ***  
LPARAMETERS group  
  
with group  
    wait window nowait .Caption  
endwith
```

event SelectItem (OldItem as Item, NewItem as Item)

Occurs when an item is selected.

Type	Description
OldItem as Item	An Item object being unselected.
NewItem as Item	An Item object being selected.

Use the SelectItem event to notify your application that a new item was selected. The [SelectItem](#) property fires the SelectItem event each time when a new item is selected. Use the [SelectGroup](#) event to notify your application that a new group was selected. Use the [Caption](#) property to get the caption of the item. Use the [Caption](#) property to get the caption of the group.

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

C#

```
private void SelectItem(object sender,exontrol.EXLISTBARLib.Item OldItem,exontrol.EXLISTBARLib.Item NewItem)
{
}
```

VB

```
Private Sub SelectItem(ByVal sender As System.Object,ByVal OldItem As exontrol.EXLISTBARLib.Item,ByVal NewItem As exontrol.EXLISTBARLib.Item)
Handles SelectItem
End Sub
```

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

C#

```
private void SelectItem(object sender,
AxEXLISTBARLib._IListBarEvents_SelectItemEvent e)
{
}
```

C++

```
void OnSelectItem(LPDISPATCH OldItem,LPDISPATCH NewItem)
{
}
```

C++ Builder

```
void __fastcall SelectItem(TObject *Sender,Exlistbarlib_tlb::IItem *OldItem,Exlistbarlib_tlb::IItem *NewItem)
{
}
```

```
}
```

Delphi

```
procedure SelectItem(ASender: TObject; OldItem : IItem;NewItem : IItem);  
begin  
end;
```

**Delphi 8
(.NET
only)**

```
procedure SelectItem(sender: System.Object; e:  
AxEXLISTBARLib._IListBarEvents_SelectItemEvent);  
begin  
end;
```

Powe...

```
begin event SelectItem(oleobject OldItem,oleobject NewItem)  
end event SelectItem
```

VB.NET

```
Private Sub SelectItem(ByVal sender As System.Object, ByVal e As  
AxEXLISTBARLib._IListBarEvents_SelectItemEvent) Handles SelectItem  
End Sub
```

VB6

```
Private Sub SelectItem(ByVal OldItem As EXLISTBARLibCtl.IItem,ByVal NewItem As  
EXLISTBARLibCtl.IItem)  
End Sub
```

VBA

```
Private Sub SelectItem(ByVal OldItem As Object,ByVal NewItem As Object)  
End Sub
```

VFP

```
LPARAMETERS OldItem,NewItem
```

Xbas...

```
PROCEDURE OnSelectItem(oListBar,OldItem,NewItem)  
RETURN
```

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

Java...

```
<SCRIPT EVENT="SelectItem(OldItem,NewItem)" LANGUAGE="JScript">  
</SCRIPT>
```

VBSc...

```
<SCRIPT LANGUAGE="VBScript">  
Function SelectItem(OldItem,NewItem)
```

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

Visual
Data...

```
Procedure OnComSelectItem Variant IIOldItem Variant IINewItem
    Forward Send OnComSelectItem IIOldItem IINewItem
End_Procedure
```

Visual
Objects

```
METHOD OCX_SelectItem(OldItem,NewItem) CLASS MainDialog
RETURN NIL
```

X++

```
void onEvent_SelectItem(COM _OldItem,COM _NewItem)
{
}
```

XBasic

```
function SelectItem as v (OldItem as OLE::Exontrol.ListBar.1::IItem,NewItem as
OLE::Exontrol.ListBar.1::IItem)
end function
```

dBASE

```
function nativeObject_SelectItem(OldItem,NewItem)
return
```

The following VB sample displays the caption of the item being selected:

```
Private Sub ListBar1_SelectItem(ByVal Item As EXLISTBARLibCtl.IItem)
    Debug.Print Item.Caption
End Sub
```

The following C++ sample displays the caption of the item being selected:

```
void OnSelectItemListbar1(LPDISPATCH Item)
{
    CItem item( Item ); item.m_bAutoRelease = FALSE;
    CString strOutput;
    strOutput.Format("%s", (LPCTSTR)item.GetCaption() );
    OutputDebugString( strOutput );
}
```

The following VB.NET sample displays the caption of the item being selected:

```
Private Sub AxListBar1_SelectItem(ByVal sender As Object, ByVal e As  
AxEXLISTBARLib._IListBarEvents_SelectItemEvent) Handles AxListBar1.SelectItem  
    With e.item  
        Debug.WriteLine(.Caption)  
    End With  
End Sub
```

The following C# sample displays the caption of the item being selected:

```
private void axListBar1_SelectItem(object sender,  
AxEXLISTBARLib._IListBarEvents_SelectItemEvent e)  
{  
    System.Diagnostics.Debug.WriteLine(e.item.Caption);  
}
```

The following VFP sample displays the caption of the item being selected:

```
*** ActiveX Control Event ***  
LPARAMETERS item  
  
with item  
    wait window nowait .Caption  
endwith
```


event SelectShortcut (OldShortcut as Variant, NewShortcut as Variant)

Fired when the user selects a new shortcut.

Type	Description
OldShortcut as Variant	A String expression that indicates the caption of the shortcut being unselected.
NewShortcut as Variant	A String expression that indicates the caption of the shortcut being selected.

The SelectShortcut event notifies your application when the user selects a shortcut. The SelectShortcut event is fired if the user clicks a shortcut in the shortcut bar, or if the code calls the [SelectShortcut](#) property. The [ShowShortcutBar](#) property shows or hides the control's shortcut bar. The [Shortcut](#) property indicates the HTML caption of the shortcut that displays the specified group. Groups with the same Shortcut property are displayed in the same shortcut. The [ShortcutPicture](#) property assigns a custom size picture to a shortcut.

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

C#private void SelectShortcut(object sender,object OldShortcut,object NewShortcut)
{
}

VBPrivate Sub SelectShortcut(ByVal sender As System.Object,ByVal OldShortcut As Object,ByVal NewShortcut As Object) Handles SelectShortcut
End Sub

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

C#private void SelectShortcut(object sender,
AxEXLISTBARLib._IListBarEvents_SelectShortcutEvent e)
{
}

C++void OnSelectShortcut(VARIANT OldShortcut,VARIANT NewShortcut)
{
}

C++ Buildervoid __fastcall SelectShortcut(TObject *Sender,Variant OldShortcut,Variant NewShortcut)

```
{  
}
```

Delphi procedure SelectShortcut(ASender: TObject; OldShortcut : OleVariant;NewShortcut : OleVariant);
begin
end;

**Delphi 8
(.NET
only)** procedure SelectShortcut(sender: System.Object; e: AxEXLISTBARLib._IListBarEvents_SelectShortcutEvent);
begin
end;

Powe... begin event SelectShortcut(any OldShortcut,any NewShortcut)
end event SelectShortcut

VB.NET Private Sub SelectShortcut(ByVal sender As System.Object, ByVal e As AxEXLISTBARLib._IListBarEvents_SelectShortcutEvent) Handles SelectShortcut
End Sub

VB6 Private Sub SelectShortcut(ByVal OldShortcut As Variant,ByVal NewShortcut As Variant)
End Sub

VBA Private Sub SelectShortcut(ByVal OldShortcut As Variant,ByVal NewShortcut As Variant)
End Sub

VFP LPARAMETERS OldShortcut,NewShortcut

Xbas... PROCEDURE OnSelectShortcut(oListBar,OldShortcut,NewShortcut)
RETURN

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

Java... <SCRIPT EVENT="SelectShortcut(OldShortcut,NewShortcut)"
LANGUAGE="JScript">

```
</SCRIPT>
```

VBSc...

```
<SCRIPT LANGUAGE="VBScript">  
Function SelectShortcut(OldShortcut,NewShortcut)  
End Function  
</SCRIPT>
```

Visual
Data...

```
Procedure OnComSelectShortcut Variant IIOldShortcut Variant IINewShortcut  
    Forward Send OnComSelectShortcut IIOldShortcut IINewShortcut  
End_Procedure
```

Visual
Objects

```
METHOD OCX_SelectShortcut(OldShortcut,NewShortcut) CLASS MainDialog  
RETURN NIL
```

X++

```
void onEvent_SelectShortcut(COMVariant _OldShortcut,COMVariant  
_NewShortcut)  
{  
}
```

XBasic

```
function SelectShortcut as v (OldShortcut as A,NewShortcut as A)  
end function
```

dBASE

```
function nativeObject_SelectShortcut(OldShortcut,NewShortcut)  
return
```