



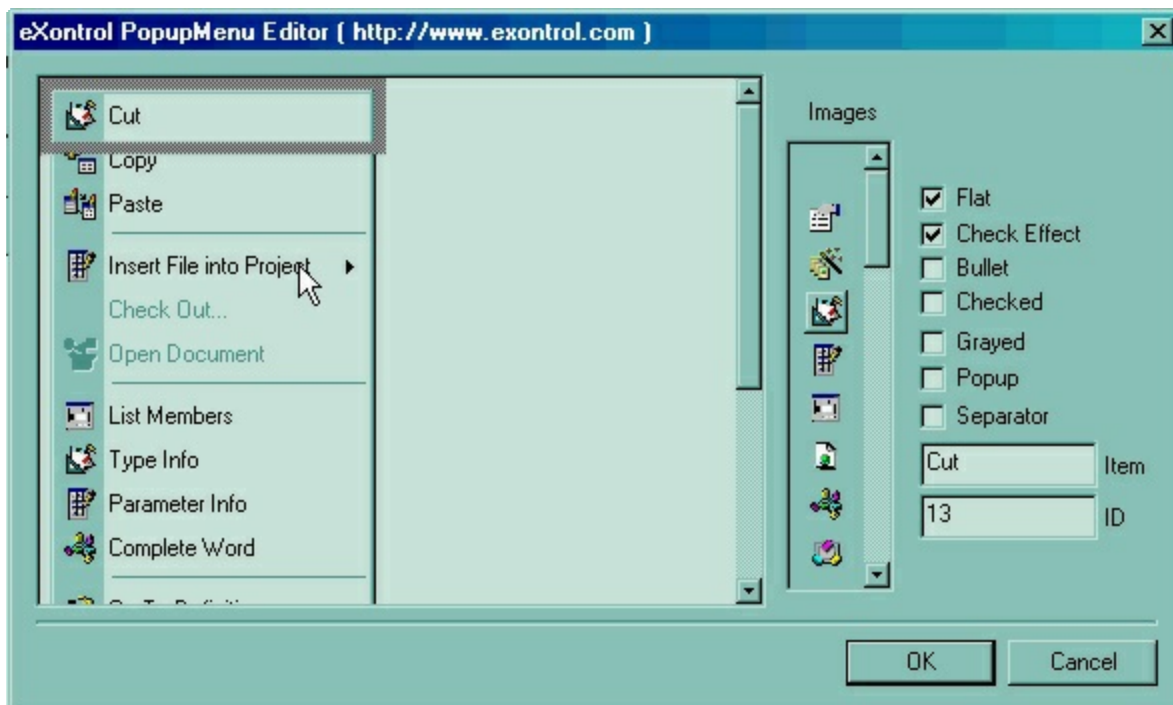
ExPopupMenu

The ExPopupMenu ActiveX control allows you to display and handle a shortcut menu (window popup menu) or a window's menu bar. The ExPopupMenu control contains also a MenuButton object that allows attaching a popup menu to a push button. Building the menu is more than intuitive because the control provides a WYSIWYG editor, at design time.

Features include:

- Ability to use **built-in HTML format** inside item
- text decorations support for HTML captions, like outlined characters, shadow, ...
- Ability to attach a popup menu to a button, **MenuButton** control
- Ability to attach a **MenuBar** to a window (form, dialog, etc)
- Easy way to handle and simulate a **drop-down button** control
- **WYSIWYG** editor that helps you to build your menu at design time
- ability to add, remove or change the items, at runtime
- ability to display images, check boxes, bullets or text as well
- ability to load icon's file or folder
- standard appearance, flat appearance, NET appearance (like in the Microsoft NET environment)

Here's a screen shot of WYSIWYG control's editor:



Ž ExPopupMenu 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 AlignEnum

Specifies the object's alignment.

Name	Value	Description
AlignCenter	1	Centers text horizontally.
AlignLeft	0	Aligns text to the left
AlignRight	2	Aligns text to the right

constants AppearanceEnum

The AppearanceEnum property specifies how the popup menu looks like. Use the [Appearance](#) property of the control to change the menu's appearance.

Name	Value	Description
Normal	0	Normal appearance.
Flat	1	Flat appearance.

Here's a screen shot of how the a "Flat" popup menu looks like:



Here's the "Normal" appearance:



constants ButtonAppearanceEnum

Specifies the control's appearance.

Name	Value	Description
None	0	No border
FlatBorder	1	Flat border
Sunken	2	Sunken border
Raised	3	Raised border
Etched	4	Etched border
Bump	5	Bump border

constants HAlignEnum

The [Show](#) and [ShowAtCursor](#) methods use the [HAlign](#) and [VAlign](#) properties.

Name	Value	Description
exCenter	4	Center. Centers the shortcut menu horizontally relative to the coordinate specified by the x parameter.
exLeft	0	Left. Positions the shortcut menu so that its left side is aligned with the coordinate specified by the x parameter.
exRight	8	Right. Positions the shortcut menu so that its right side is aligned with the coordinate specified by the x parameter.

constants MenuAlignEnum

Specifies the alignment of the menu for a MenuButton object.

Name	Value	Description
MenuLeft	0	MenuLeft.
MenuRight	1	MenuRight.
MenuTop	2	MenuTop.
MenuBottom	3	MenuBottom.
MenuCenter	4	MenuCenter.
MenuBottomRight	5	MenuBottomRight.
MenuTopRight	6	MenuTopRight.

constants MenuButtonTypeEnum

Specifies the type of a MenuButton object.

Name	Value	Description
Fixed	0	Fixed.
PushButton	1	PushButton.

constants ItemTypeEnum

The ItemTypeEnum defines the type of the [Item](#) object. Use the [Add](#) property to add an item of a specified type.

Name	Value	Description
Default	0	(Default)
Separator	1	An item of separator type.
SubMenu	2	An item that contains a sub menu.
ShortCut	3	Adds an item of shortcut type. If the Caption points to a folder a new item of SubMenu type is added.
ShortCutFolder	4	Adds an item of shortcut type. If the Caption points to a folder a new item of Default type is added.

constants VAlignEnum

Specifies the menu's alignment relative to the cursor position.

Name	Value	Description
exVCenter	16	Centers the shortcut menu vertically relative to the coordinate specified by the y parameter.
exBottom	32	Positions the shortcut menu so that its bottom side is aligned with the coordinate specified by the y parameter.
exTop	0	Positions the shortcut menu so that its top side is aligned with the coordinate specified by the y parameter.

Command object

A Command object holds information about an item being selected.

Name	Description
Caption	Retrieves or sets a value that indicates the caption of item menu.
Checked	Retrieves or sets a value that indicates if the item menu is checked or unchecked.
FilePath	Retrieves a value that indicates the file path of the command, when it is of ShortCut type.
Grayed	Retrieves or sets a value that indicates wehter the item menu is enabled or disabled.
ID	Retrieves the item menu command identifier.
Image	Retrieves or sets the image associated with the item menu,

property Command.Caption as String

Retrieves or sets a value that indicates the caption of the item menu.

Type	Description
String	A string expression that indicates the caption of the item menu.

Use Caption property to get the command's caption that user has selected. Use the [Command](#) property of the control to get a [Command](#) object based on the item's identifier. You can use [Caption](#) property of [Item](#) object to retrieves the items' caption. The Caption property may include built-in HTML tags like follows:

- ** 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[:width]** inserts an icon inside the cell's caption. The number indicates the index of the icon being inserted in the caption.
- **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.

The following sample shows how to get the caption of the menu item that has been selected on the popup menu:

```
Private Sub Form_MouseUp(Button As Integer, Shift As Integer, X As Single, Y As Single)
    If (Button = 2) Then
        Dim nID As Long
        nID = PopupMenu1.ShowAtCursor
        If (nID > 0) Then
            MsgBox "You have selected the '" & PopupMenu1.Command(nID).Caption & "'"
```

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

property Command.Checked as Boolean

Retrieves or sets a value that indicates if the item menu is checked or unchecked.

Type	Description
Boolean	A boolean expression that indicates if the item menu is checked or unchecked.

Use the Checked property to check or uncheck a menu identifier. Use the [Command](#) property of the control to get a [Command](#) object based on the item's identifier. The following sample shows how to check/uncheck an item:

```
Private Sub Form_MouseUp(Button As Integer, Shift As Integer, X As Single, Y As Single)
    If (Button = 2) Then
        ' Adds a new item to menu with the identifier 1234
        Dim it As Item
        Set it = PopupMenu1.Items.Item("Auto saving")
        If it Is Nothing Then Set it = PopupMenu1.Items.Add("Auto saving", Default, 1234)
        ' Displays the context menu
        Dim nID As Long
        nID = PopupMenu1.ShowAtCursor
        ' Checks or unchecks the item
        If nID = 1234 Then
            it.Check = Not it.Check
        End If
    End If
End Sub
```

property Command.FilePath as String

Retrieves a value that indicates the file path for an item of ShortCut type.

Type	Description
String	A string expression that indicates the file path.

The following sample shows how to run the application when an item of ShortCut type is selected:

```
Private Declare Function ShellExecute Lib "shell32.dll" Alias "ShellExecuteA" (ByVal hwnd As Long, ByVal lpOperation As String, ByVal lpFile As String, ByVal lpParameters As String, ByVal lpDirectory As String, ByVal nShowCmd As Long) As Long
Private Const SW_SHOW = 5

Private Sub Form_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
    Dim i As Long
    i = PopupMenu1.ShowAtCursor
    If (i >= 1234) Then
        ' Is Internet Explorer?
        If i = 1234 Then
            ' You can call ShellExecute API and you can pass argumets!
            ShellExecute 0, "open", PopupMenu1.Command(i).FilePath,
"https://www.exontrol.com", "", SW_SHOW
        Else
            ' You can call ShellExecute API and you can pass argumets!
            ShellExecute 0, "open", PopupMenu1.Command(i).FilePath, "", "", SW_SHOW
        End If
    End If
End Sub
```

property Command.Grayered as Boolean

Retrieves or sets a value that indicates whether the item menu is enabled or disabled.

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

Use the Grayered property to enable or disable an item. the [Command](#) property of the control to get a [Command](#) object based on the item's identifier. The following sample shows how to disable an item:

```
PopupMenu1.Command(1234).Grayered = True
```

The following sample shows how to add an disabled item:

```
With PopupMenu1.Items
.Add("Item", Default, 1234).Enabled = True
End With
```


property Command.ID as Long

Retrieves the item menu command identifier.

Type	Description
Long	A long expression that indicates the menu command identifier.

Use the [Command](#) property of the control to get a [Command](#) object based on the item's identifier. You can use [Caption](#) property of the [Item](#) object to retrieves the items' caption. Use the [Add](#) property of the [Menu](#) object to add a new item to your context menu. Use the [ID](#) property of the Item object to change the item's identifier. It is not recommended changing the identifier at runtime. The ID should be unique, but it is not a requirement. If you have two or more items with the same identifier the Command property will get you the first command that has that identifier. The following sample shows how to get the caption of the menu item that has been selected on the popup menu:



```
Private Sub Form_MouseUp(Button As Integer, Shift As Integer, X As Single, Y As Single)
    If (Button = 2) Then
        Dim nID As Long
        nID = PopupMenu1.ShowAtCursor
        If (nID > 0) Then
            MsgBox "You have selected the '" & PopupMenu1.Command(nID).Caption & "'"
        End If
    End If
End Sub
```

property Command.Image as Long

Retrieves or sets the image associated with the item menu

Type	Description
Long	A long expression that indicates the image associated with the item menu.

Use [Images](#) method of the control to associate at runtime an images list to your popup menu. Use the Image property of the Command to change the image. Use the Image() = -1 to remove the item's image.

The following sample shows how to use your own images to simulate a check box item (add two images to control's images list at the top of the list in the following order  )

```
Private Sub Form_MouseUp(Button As Integer, Shift As Integer, X As Single, Y As Single)
    If (Button = 2) Then
        ' Adds a new item to menu with the identifier 1234
        Dim it As Item
        Set it = PopupMenu1.Items.Item("Auto saving")
        If it Is Nothing Then
            Set it = PopupMenu1.Items.Add("Auto saving", Default, 1234)
            it.Image = 0
        End If
        ' Displays the context menu
        Dim nID As Long
        nID = PopupMenu1.ShowAtCursor
        ' Changes the item's image if the user has selected the 1234 command menu
        If nID = 1234 Then
            it.Image = (it.Image + 1) Mod 2
        End If
    End If
End Sub
```

Item object

The Item object represents an menu item at runtime. Use Item object when you need to add new items to the menu. Use the [Item](#) property of the Menu object to access an Item object. The Item object contains the following information about menu's item:

Name	Description
Bullet	Retrieves or sets a value indicating whether the item is of bullet type.
Caption	Retrieves or sets a value that indicates the item's caption.
Check	Retrieves or sets a value that indicates whether the item is of check type.
Enabled	Specifies whether the item is enabled or disabled.
FilePath	Retrieves a value that indicates the file path of the item, when it is of ShortCut type.
Gray	Specifies whether a disabled item appears as grayed.
Highlight	Specifies whether the focused item appears selected.
HTMLImage	Retrieves or sets a value that indicates the key of the image (HTMLPicture method) to be displayed on the item (left side).
ID	Retrieves or sets a value that indicates the item's identifier.
Image	Retrieves or sets a value that indicates the item's index image.
ItemData	Associates an extra data to an item.
Parent	Gets the parent item.
Separator	Retrieves or sets a value that indicates whether the source is a separator item.
SubMenu	Retrieves a Menu object that indicates the item's sub menu.
Visible	Specifies whether the item is visible or hidden.

property Item.Bullet as Boolean

Retrieves or sets a value indicating whether the item is of bullet type.

Type	Description
Boolean	A boolean expression indicating whether the item is of bullet type.

Use [Check](#) property to check or uncheck an item. The following sample shows to use bullets in your application:

```
Private Sub Form_Load()  
    PopupMenu1.Items.Add("Bullet 1", Default, 1).Bullet = True  
    PopupMenu1.Items.Add("Bullet 2", Default, 2).Bullet = True  
    PopupMenu1.Items.Add("Bullet 3", Default, 3).Bullet = True  
    PopupMenu1.Items(1).Check = True  
End Sub  
  
Private Sub Form_MouseUp(Button As Integer, Shift As Integer, X As Single, Y As Single)  
    If (Button = 2) Then  
        Dim nID As Long  
        nID = PopupMenu1.ShowAtCursor  
        For i = 1 To 3  
            PopupMenu1.Command(i).Checked = False  
        Next  
        PopupMenu1.Command(nID).Checked = True  
    End If  
End Sub
```

property Item.Caption as String

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

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

Use the [Command](#) property of the control to get an command based on the item's identifier. Use the [Add](#) property of the Menu object to add a new item. Use the [Items](#) property of the control to access to menu items at runtime. The Caption property supports built-in HTML format like follows:

- ** 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[:width]** inserts an icon inside the cell's caption. The number indicates the index of the icon being inserted in the caption.
- **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 sample shows adds two items, one of separator type, and one of default type:

```
With PopupMenu1.Items
.Add "", Separator
```

```
Dim it As Item
Set it = .Add("Item")
it.ID = 1234
it.Image = 1
End With
```

property Item.Check as Boolean

Retrieves or sets a value that indicates whether the item is of check type.

Type	Description
Boolean	A boolean expression that indicates the item's check state.

Use the [Bullet](#) property to change the type of the item. Use the [Command](#) property of the control to get an command based on the item's identifier. Use the [Add](#) property of the Menu object to add a new item. Use the [Items](#) property of the control to access to menu items at runtime. The following sample shows how to add two items, one of separator type, and one of default type:

```
With PopupMenu1.Items
    .Add "", Separator
    With .Add("Item")
        .ID = 1234
        .Image = 1
        .Check = True
    End With
End With
```


property Item.Enabled as Boolean

Retrieves or sets a value indicating whether the item is enabled or disabled.

Type	Description
Boolean	A boolean expression indicating whether the item is enabled or disabled.

Use the Enabled property to enable or disable an item. The following sample shows how to add at runtime a disabled item:

```
With PopupMenu1.Items
.Add("Item", Default, 1234).Enabled = False
End With
```

The following sample shows how to disable an item after it was selected:

```
Private Sub Form_MouseUp(Button As Integer, Shift As Integer, X As Single, Y As Single)
    If (Button = 2) Then
        Dim nID As Long
        nID = PopupMenu1.ShowAtCursor
        If (nID = 1234) Then
            PopupMenu1.Command(1234).Grayed = True
        End If
    End If
End Sub
```

property Item.FilePath as String

Retrieves a value that indicates the file path of an item of ShortCut type.

Type	Description
String	A string expression that indicates the the file path.

The following sample shows how to add an item of ShortCut type to your menu:

```
With PopupMenu1.Items
Dim i As Item
Set i = .Add("C:\Documents and Settings\All Users\Start Menu\Programs", ShortCut)
i.SubMenu.Add("C:\Program Files\Internet Explorer\IEXPLORE.EXE", ShortCut,
1234).Caption = "Internet Explorer"
i.SubMenu.Add("C:\WINNT\explorer.exe", ShortCut, 1235).Caption = "Windows Explorer"
```

The following sample shows how to add your Notepad application to menu:

```
PopupMenu1.Add "c:\winnt\system32\notepad.exe", ShortCut, 1236
```

The following sample shows how to run the application when a item of ShortCut type is selected:

```
Private Declare Function ShellExecute Lib "shell32.dll" Alias "ShellExecuteA" (ByVal hwnd
As Long, ByVal lpOperation As String, ByVal lpFile As String, ByVal lpParameters As String,
ByVal lpDirectory As String, ByVal nShowCmd As Long) As Long
Private Const SW_SHOW = 5
```

```
Private Sub Form_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
Dim i As Long
i = PopupMenu1.ShowAtCursor
If (i >= 1234) Then
' Is Internet Explorer?
If i = 1234 Then
' You can call ShellExecute API and you can pass argumets ( in this case we pass
"https://www.exontrol.com" for Internet Explorer item!
ShellExecute 0, "open", PopupMenu1.Command(i).FilePath,
"https://www.exontrol.com", "", SW_SHOW
Else
```

' You can call ShellExecute API and you can pass arguments!

ShellExecute 0, "open", PopupMenu1.Command(i).FilePath, "", "", SW_SHOW

End If

End If

End Sub

property Item.Gray as Boolean

Specifies whether a disabled item appears as grayed.

Type	Description
Boolean	A boolean expression that indicates whether a disabled item appears as grayed.

The Gray property has effect only if [Enabled](#) property is False. Use the Gray and Highlight properties to specify how the a disabled item looks like.

property Item.Highlight as Boolean

Specifies whether the focused item appears selected.

Type	Description
Boolean	A boolean expression that specifies whether the focused item appears selected.

Use the Highlight property to specify whether the focused item appears selected. By default, the Highlight property is True. The [Gray](#) and Highlight properties of Item object helps you to specify how the item appear when it is disabled and selected. For instance, the following sample adds an un selectable item:

```
With PopupMenu1
  With .Items.Add("My <b>title</b>", Default, 1234)
    .Highlight = False
    .Enabled = False
    .Gray = False
  End With
End With
```

property Item.HTMLImage as String

Retrieves or sets a value that indicates the key of the image (HTMLPicture method) to be displayed on the item (left side).

Type	Description
String	A String expression that indicates the key of the picture to be displayed on the left side of the caption.

The HTMLImage property assigns a picture to the left side of the caption. The key of the picture to be displayed must be loaded previously using the [HTMLPicture](#) property. The HTMLImage property has effect only if the Image property is -1 (by default). Use the [Image](#) property to assign an icon from the [Images](#) collection to the left side of the caption.

The following VFP samples loads the picture using the HTMLPicture method, and displays it on the left side of the caption using the HTMLImage property.

```
popupMenu = CreateObject("Exontrol.PopupMenu")
with popupMenu
.HTMLPicture("pic1") = "C:\exontrol\images\colorize.gif"
.Items.ToString = "Item A[himg=pic1]"
iShowAtCursor = .ShowAtCursor()
IF ( iShowAtCursor # 0 ) then
?( .Items.item(iShowAtCursor).Caption )
ENDIF
endwith
```

or:

```
popupMenu = CreateObject("Exontrol.PopupMenu")
with popupMenu
.HTMLPicture("pic1") = "C:\exontrol\images\colorize.gif"
.Items.Add("Item A").HTMLImage = "pic1"
iShowAtCursor = .ShowAtCursor()
IF ( iShowAtCursor # 0 ) then
?( .Items.item(iShowAtCursor).Caption )
ENDIF
endwith
```

These two samples are equivalent.

property Item.ID as Long

Retrieves or sets a value that indicates the item's identifier.

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

It is recommended to use unique identifiers for items. If the menu contains more items with the same identifier the [Command](#) property retrieves the first occurrence. Use the [Add](#) property to add a new item to the menu. You can associate an identifier to an item using the Add property of the [Menu](#) object. Use the [Items](#) property of the control to access the control menu items. The following sample shows how to add a new item of bullet type:

```
With PopupMenu1.Items
.Add("Item", Default, 1234).Bullet = True
End With
```

property Item.Image as Long

Retrieves or sets a value that indicates the item's index image.

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

By default, when a new item is added to the menu no image is associated. Use the Image property to associate an image to an item.

The following sample shows how to add an item:

```
With PopupMenu1.Items
.Add("Auto saving", Default, 1234).Image = 0
End With
```

The following sample shows how to simulate an item of check type using your own images:

```
Private Sub Form_MouseUp(Button As Integer, Shift As Integer, X As Single, Y As Single)
    If (Button = 2) Then
        Dim nID As Long
        nID = PopupMenu1.ShowAtCursor
        If (nID = 1234) Then
            PopupMenu1.Command(1234).Image = (PopupMenu1.Command(1234).Image +
1) Mod 2
        End If
    End If
End Sub
```


property Item.ItemData as Variant

Associates an extra data to an item.

Type	Description
Variant	A Variant expression associated to the item

Use the ItemData property to associate an extra data to an item. The ItemData could be an object, a string, a date, a long expression, a double expression, or anything that a Variant structure can hold. The ItemData gets or sets the user-definable data for the current item. Use the [Item](#) property to access an item object giving its identifier. Use the [ItemByData](#) property to search for an item giving its associated data.

property Item.Parent as Item

Gets the parent item.

Type	Description
Item	An Item object that identifies the parent's item.

The Parent property gets the parent item. The Parent property gets nothing if there is no parent item.

property Item.Separator as Boolean

Retrieves or sets a value that indicates whether the source is a separator item.

Type	Description
Boolean	A boolean expression that indicates whether the source is a separator item.

Use [Add](#) property of the [Menu](#) object to add a new item to the menu. The following sample shows how to add a new separator item to the menu:

```
With PopupMenu1.Items
.Add "", Separator
End With
```

property Item.SubMenu as Menu

Retrieves a Menu object that indicates the item's sub menu.

Type	Description
Menu	A boolean expression that indicates the item's sub menu.

Use the SubMenu property to add sub menus to a menu. The SubMenu property retrieves nothing if the item contains no sub menu. An item contains a sub menu if it was created by the [Add](#)(, [SubMenu](#)).

The following sample shows how to add an item of SubMenu type:

```
Dim m As EXPOPUPMENULibCtl.Menu
Set m = PopupMenu1.Items.Add("SubMenu", SubMenu).SubMenu
m.Add "Item 1", Default, 1
m.Add "Item 2", Default, 2
m.Add "Item 3", Default, 3
```

property Item.Visible as Boolean

Specifies whether the item is visible or hidden.

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

By default, the Visible property is True, and that means any added item is visible. Use the Visible property to hide or show items at runtime, instead removing and adding them later. Use the [Remove](#) method to remove an item. Use the [Add](#) method to add a new item.

Menu object

Using the Menu object you can add, remove or change menu items. A Menu object contains a collection of [Item](#) objects. Use the [Items](#) property of the control to access the menu items of the control. Use the [SubMenu](#) property of the Item object to access the item's sub menu.

Name	Description
Add	Adds a new item to menu and retrieves the newly created object.
Clear	Clear the menu items.
Count	Counts the items.
Item	Returns a specific Item object given its caption or its identifier.
Remove	Removes the item given its name or its identifier.
ToString	Saves or loads the menu from a formatted string.

method Menu.Add (Caption as String, [ItemType as ItemTypeEnum], [ID as Variant])

Adds a new item to menu and retrieves the newly created object.

Type	Description
Caption as String	A string expression that indicates the item's caption or the file path if the ItemType is ShortCut.
ItemType as ItemTypeEnum	An ItemTypeEnum expression that indicates the item's type.
ID as Variant	A long expression that indicates the item's identifier.
Return	Description
Item	An Item object that represents the newly created item.

The Caption parameter supports built-in HTML like described in the [Caption](#) property. Use the Add property to add items to the menu. Use [Clear](#) method to clear your popup menu. Use [SubMenu](#) property to access the item's sub menu. Add method adds an icon to Images collection that corespond to file path, if the newly item created is of ShortCut type.. Use the [Remove](#) method to remove an item. Use the [Visible](#) property to show or hide an item.

The following sample shows how to add an item:

```
PopupMenu1.Items.Add "Item", Default, 1234
```

The following sample shows how to add a new separator item:

```
PopupMenu1.Items.Add "", Separator
```

The following sample shows how to add item to a submenu:

```
Dim m As EXPOPUPMENULibCtl.Menu
Set m = PopupMenu1.Items.Add("SubMenu", SubMenu).SubMenu
m.Add "Item 1", Default, 1
m.Add "Item 2", Default, 2
m.Add "Item 3", Default, 3
```

The following sample shows how to add an item of ShortCut type to your menu:

```
Dim i As Item
' Adds a new item of SubMenu type that points to a Folder.
```

```
Set i = .Add("C:\Documents and Settings\All Users\Start Menu\Programs", ShortCut)
```

```
' Adds shortcuts to files.
```

```
i.SubMenu.Add("C:\Program Files\Internet Explorer\IEXPLORE.EXE", ShortCut,  
1234).Caption = "Internet Explorer"
```

```
i.SubMenu.Add("C:\WINNT\explorer.exe", ShortCut, 1235).Caption = "Windows Explorer"
```

```
.Add "c:\winnt\system32\notepad.exe", ShortCut, 1236
```

```
' Adds a shortcut to a folder. The newly created item contains no submenu, and needs an  
identifier
```

```
.Add "C:\Documents and Settings\All Users\Start Menu\Programs", ShortCutFolder, 1237
```

The following sample shows how to add your Notepad application to menu:

```
PopupMenu1.Add "c:\winnt\system32\notepad.exe", ShortCut, 1236
```

The following sample shows how to run the application when a item of ShortCut type is selected:

```
Private Declare Function ShellExecute Lib "shell32.dll" Alias "ShellExecuteA" (ByVal hwnd  
As Long, ByVal lpOperation As String, ByVal lpFile As String, ByVal lpParameters As String,  
ByVal lpDirectory As String, ByVal nShowCmd As Long) As Long
```

```
Private Const SW_SHOW = 5
```

```
Private Sub Form_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
```

```
Dim i As Long
```

```
i = PopupMenu1.ShowAtCursor
```

```
If (i >= 1234) Then
```

```
    ' Is Internet Explorer?
```

```
    If i = 1234 Then
```

```
        ' You can call ShellExecute API and you can pass arguments ( in this case we pass
```

```
"https://www.exontrol.com" for Internet Explorer item!
```

```
        ShellExecute 0, "open", PopupMenu1.Command(i).FilePath,
```

```
"https://www.exontrol.com", "", SW_SHOW
```

```
    Else
```

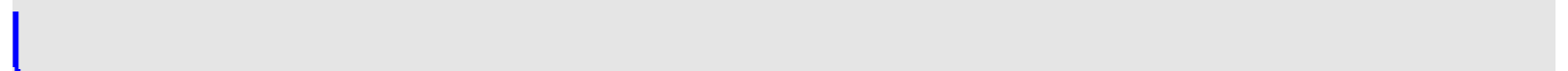
```
        ' You can call ShellExecute API and you can pass arguments!
```

```
        ShellExecute 0, "open", PopupMenu1.Command(i).FilePath, "", "", SW_SHOW
```

```
    End If
```

```
End If
```

```
End Sub
```

method Menu.Clear ()

Clear the menu items.

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

Use the [Remove](#) method to remove an item.

property Menu.Count as Long

Counts the items.

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

Use the Count property to count the items within the collection.

property Menu.Item (ID as Variant) as Item

Returns a specific Item object given its caption or its identifier.

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

Use the Item property to access to a Item object. Use the [Caption](#) property to specify the item's caption. Use the [Visible](#) property to specify whether the item is visible or hidden. Use the [Item](#) property of the PopupMenu control to retrieves the item giving its identifier. The [ID](#) property specifies the identifier of the item. Use the [ItemByData](#) property to search for an item giving its associated data ([ItemData](#) property)

The Item property is the default property of the menu object so the following statements are equivalents:

```
PopupMenu1.Items.Item ("Item 1")
PopupMenu1.Items ("Item 1")
```

method Menu.Remove (ID as Variant)

Removes the item given its caption or its identifier.

Type	Description
ID as Variant	A long expression that indicates the item's identifier or a string expression that indicates the item's caption.

Use the [Clear](#) method to remove all items of the Menu object. Use the [ID](#) property to specify the item's identifier. Use the [Caption](#) property to specify the item's caption. Use the [Visible](#) property to specify whether the item is visible or hidden.

property Menu.ToString as String

Saves or loads the menu from a formatted string.

Type	Description
String	A String expression that indicates the list of items.

Use the ToString method to quick load the items from a formatted string.

The ToString's syntax in BNF notation:

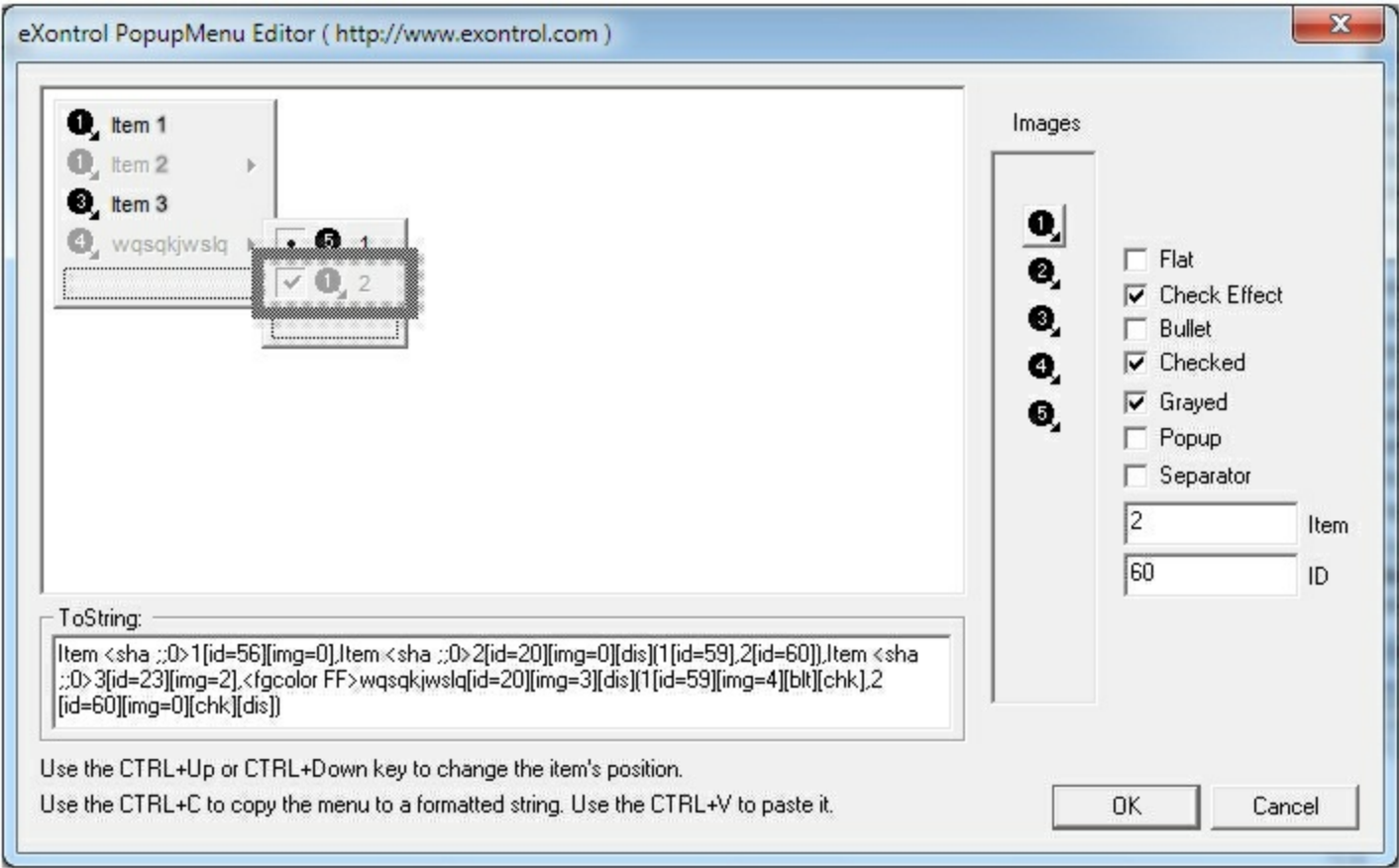
```
<TOSTRING> ::= <ITEMS>
<ITEMS> ::= <ITEM>["("<ITEMS>")"][","<ITEMS>]
<ITEM> ::= <NAME>[<OPTIONS>]
<OPTIONS> ::= "["<OPTION>"]"["["<OPTIONS>"]"]
<OPTION> ::= <PROPERTY>["="<VALUE>]
<PROPERTY> ::= "id"|"img"|"sep"|"blt"|"chk"|"dis"|"bld"
```

where the <NAME> is the name of the item. The <VALUE> tag is valid for the options "img","id","bg","fg" and "edit", like follows:

- The <VALUE> tag for "img" option is an integer expression, that indicates the index of the icon being displayed for the item.
- The <VALUE> tag for "id" option is an integer expression, that indicates the identifier of the item.
- The <VALUE> tag for "bg" or "fg" option is a color expression, that indicates the color of the item. The <VALUE> could be a RGB expression (RGB(RR,GG,BB), where RR is the red value, the GG is the green value, and the BB is the blue value), or a long expression.
- The <VALUE> tag for "edit" option is a string expression (without " characters), that indicates the caption of the edit inside the edit control.
- If the "sep" option is present, the item is a separator item.
- If the "blt" option is present, the item displays a bullet.
- If the "chk" option is present, the item displays a check box.
- If the "dis" option is present, the item is disabled.

Is there any way to get this formatted string from the control's editor? Open the control in design mode, select the Properties item in its context menu, and locate the Editor page, where you find the "Invoke Internal Editor" button, click it, and the control's Editor page is opened. Build your menu. Click an item, so it becomes focused, and press the CTRL + C, a beep should be hear. Now, open a notepad editor, and press CTRL + V, you should get the formatted string for ToString method. Click the designer, so no item is focused, and press the CTRL + C key, and so the entire menu is copied to the clipboard, as ToString syntax.

Here's how the WYSWYG editor looks like:



MenuButton 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: {3AD387DD-2DBF-4C11-8591-9DE1E65C28C7}. The object's program identifier is: "ExPopupMenu.MenuButton". The /COM object module is: "ExPMenu.dll"

A MenuButton object displays a [PopupMenu](#) object when user clicks a drop down button.
The MenuButton object

Name	Description
Alignment	Retrieves or sets the caption's alignment.
Appearance	Retrieves or sets the menu button's appearance.
BackColor	Retrieves or sets the control's background color.
Caption	Retrieves or sets the control's caption.
CursorInside	Checks whether the cursor is over the control.
Dismiss	Closes the popup menu if it is opened.
DropDownVisible	Retrieves or sets a value indicating whether the control contains a drop down button.
Enabled	Enables or disables the control.
EventParam	Retrieves or sets a value that indicates the current's event parameter.
Font	Retrieves or sets the control's font.
ForeColor	Retrieves or sets the control's foreground color.
hWnd	Retrieves the control's window handle.
LastID	Retrieves the last command identifier that was selected.
Menu	Associates a menu to a button.
MenuAlign	Retrieves or sets the menu's alignment.
Picture	Retrieves or sets the control's picture.
PictureAlignment	Retrieves or sets a value that indicates the picture's alignment.
Push	Presses the button.
ShowFocusRect	Retrieves or sets a value that indicates whether the focus rectangle is visible or hidden.
Type	Retrieves or sets the control's type.
Version	Retrieves the control's version.

property MenuButton.Alignment as AlignEnum

Retrieves or sets the caption's alignment.

Type	Description
AlignEnum	An AlignEnum expression that indicates the alignment of the control's caption.

Use the [Caption](#) property to assign a caption to the control. Use the Alignment property aligns the control's caption.

property MenuButton.Appearance as ButtonAppearanceEnum

Retrieves or sets the menu button's appearance.

Type	Description
ButtonAppearanceEnum	A ButtonAppearanceEnum expression that indicates the control's appearance.

The Appearance property specifies the control's appearance.

property MenuButton.BackColor as Color

Retrieves or sets the control's background color.

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

Use the BackColor and [ForeColor](#) properties to specify the control background and foreground colors.

property MenuButton.Caption as String

Retrieves or sets the control's caption.

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

The Caption property specifies the control's caption. Use the Caption property to change the control's caption.

property MenuButton.CursorInside as Boolean

Checks whether the cursor is over the control.

Type	Description
Boolean	A boolean expression that indicates whether the cursor is inside of the control.

Use the CursorInside property to check whether the cursor is over a MenuButton object.

method **MenuButton.Dismiss ()**

Closes the popup menu if it is opened.

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

Closes the assigned menu by code.

property MenuButton.DropDownVisible as Boolean

Retrieves or sets a value indicating whether the control contains a drop down button.

Type	Description
Boolean	A boolean expression that indicates whether the control displays a drop down button.

Use the DropDownVisible property to specify when the control displays a drop down button. By default, the DropDownVisible property is True.

property MenuButton.Enabled as Boolean

Enables or disables the control.

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

By default, the control is enabled. Use the Enabled property to enable or disable the control.

property MenuButton.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.

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

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

property MenuButton.Font as IFontDisp

Retrieves or sets the control's font.

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

Use the Font property to specify the MenuButton's font.

property MenuButton.ForeColor as Color

Retrieves or sets the control's foreground color.

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

Use the [BackColor](#) and ForeColor properties to specify the control background and foreground colors.

property MenuButton.hWnd as Long

Retrieves the control's window handle.

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

Use the hWnd property to access the handle of the control's window.

property MenuButton.LastID as Long

Retrieves the last command identifier that was selected.

Type	Description
Long	A long expression that indicates the last identifier being selected.

The LastID property gets the identifier of the item being clicked.

property MenuButton.Menu as PopupMenu

Associates a menu to a button.

Type	Description
PopupMenu	Assigns a PopupMenu to a MenuButton object

Use the Menu property to assign a PopupMenu object to the MenuButton object.

The following sample assigns a drop down menu (a PopupMenu) to the MenuButton object:

```
With MenuButton1
    Set .Menu = PopupMenu1
End With
```

property MenuButton.MenuAlign as MenuAlignEnum

Retrieves or sets the menu's alignment.

Type	Description
MenuAlignEnum	A MenuAlignEnum expression that indicates the menu's alignment.

Aligns the menu assigned to the MenuButton object.

property MenuButton.Picture as Variant

Retrieves or sets the control's picture.

Type	Description
Variant	A Picture object that indicates the control's background picture, or a string expression that indicates the file name of the picture file.

Use the Picture method to load a picture on the control's background. Use the [PictureAlignment](#) to align the control's picture. The following sample loads a picture on the control's background:

```
MenuButton1.Picture = "c:\winnt\zapotec.bmp"
```


property MenuButton.PictureAlignment as AlignEnum

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

Type	Description
AlignEnum	An AlignEnum expression that indicates the alignment of the control's picture.

The PictureAlignment aligns the control's picture. If the control has no picture attached the PictureAlignment property has no effect.

method MenuButton.Push ()

Presses the button.

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

The Push and [Dismiss](#) methods opens or closes the assigned menu.

property MenuButton.ShowFocusRect as Boolean

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

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

By default, the ShowFocusRect property is True. Use the ShowFocusRect property to hide the focus rectangle that's painted when the control has the focus.

property MenuButton.Type as MenuButtonTypeEnum

Retrieves or sets the control's type.

Type	Description
MenuButtonTypeEnum	A MenuButtonTypeEnum expression that indicates the type of the control.

Specifies the type of the MenuButton object.

property MenuButton.Version as String

Retrieves the control's version.

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

The control's Version property gets the version of the control.

PopupMenu 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: {462D5053-2D60-4022-B583-7E34AA0F90B7}. The object's program identifier is: "Exontrol.ExPopupMenu". The /COM object module is: "ExPMenu.dll"

The ExPopupMenu ActiveX control allows you to display and handle a shortcut menu (window popup menu). Many times your application needs to provide a shortcut menu based on a context. Well, the ExPopupMenu is easy to use, easy to integrate into your application, and fits all your requirments. All that you need to do is to build your menu and to invoke Show property. The control provides a WYSIWYG editor that helps you to build your menu at design time. Only few clicks an you have a popup menu that contains images. The control provides few COM objects that helps user to add, remove or change the items, at runtime. A menu item can display images, check boxes, bullets or text. The menu's appearance can be changed to flat, to let you menu looks like popup menus in Microsoft NET environment. The ExPopupMenu ActiveX Control exports the following properties:

Name	Description
Appearance	Retrieves or sets a value that indicates the control's appearance.
Attach	Changes the window's menu bar.
BackColor	Retrieves or sets the control's background color.
CloseOnClick	Specifies whether the control is closed when user clicks.
Command	Retrieves a Command object based on menu item identifier.
Debug	Retrieves or sets a value that indicating whether the item's identifier is visible.
Detach	Detaches the window's menu, attached by the Attach method.
EventParam	Retrieves or sets a value that indicates the current's event parameter.
FlatImageWidth	Specifies the width of the column to display the icons/images when the control's MenuAppearance is exMenuFlat.
Font	Retrieves or sets the control's font.
ForeColor	Retrieves or sets a value that indicates the control's foreground color.
HAlign	Specify how the control positions the shortcut menu horizontally.

Hide	Hides the control.
HTMLPicture	Adds or replaces a picture in HTML captions.
hWnd	Retrieves the control window's handle.
Images	Sets the control's image list at runtime. The Handle should be a handle to an Image List control.
Item	Returns a specific Item object given its caption or its identifier.
ItemByData	Searches for a specific Item object given its associated data.
ItemHeight	Specifies the item's height in pixels.
Items	Retrieves a Menu object to handle adding, removing or changing items at runtime.
SelBackColor	Retrieves or sets a color that indicates the selection's background color.
SelForeColor	Retrieves or sets a color that indicates the selection's text color.
ShadowColor	Specifies the shadow color.
Show	Displays the popup menu at given coordinates, and retrieves the menu command identifier. The coordinates should be relative to container coordinates.
ShowAtCursor	Shows the popup menu at current cursor position
ShowAtWindow	Displays the popup menu relative to the given window.
VAlign	Specify how the control positions the shortcut menu vertically.
Version	Retrieves the control's version.

property PopupMenu.Appearance as AppearanceEnum

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

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

The menu's appearance can be selected by the user either using the container browser or using the ExPopupMenu's editor. The editor of the ExPopupMenu can be called by selecting the Properties context menu at design time.

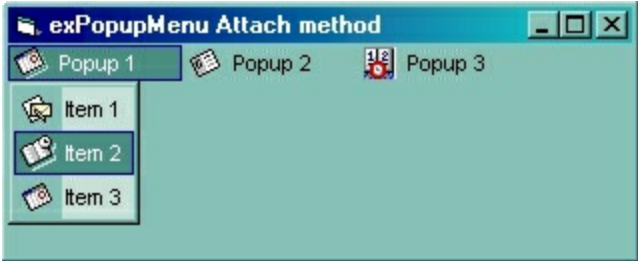
method PopupMenu.Attach (hWnd as Long)

Changes the window's menu bar.

Type	Description
hWnd as Long	A long expression that indicates the handle of the window

Use the Attach method to attach a menu to a window. The control fires the [Command](#) event when the user selects an item. Use the [Detach](#) method to remove the window's menu bar. If you are calling this property in VB, you need to set the form's NegotiateMenus property on False. Use the [Show](#) or [ShowAtCursor](#) property to display the control as a context menu.

The following screen show shows the menu when calling the Attach method:



property PopupMenu.BackColor as Color

Retrieves or sets the control's background color.

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

Use the BackColor property to change the control's background color. Use the [ForeColor](#) property to change the control's foreground color.

property PopupMenu.CloseOnClick as Boolean

Specifies whether the control is closed when user clicks.

Type	Description
Boolean	A boolean expression that indicates whether the control is closed when user clicks the menu.

Use the CloseOnClick event to specify whether the menu is closed when user clicks the menu.

property PopupMenu.Command (ID as Long) as Command

Retrieves a Command object based on menu item identifier.

Type	Description
ID as Long	A long expression that indicates the command's identifier.
Command	A Command object that points to an menu item.

Use the Command property to get information about the selected item.

The following sample displays the caption of the selected item:

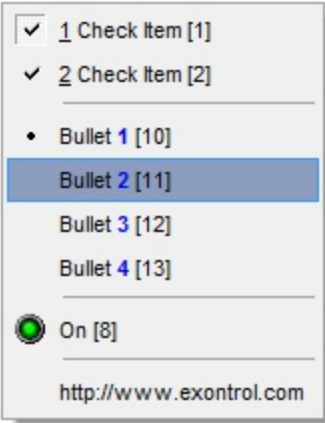
```
Private Sub Form_MouseUp(Button As Integer, Shift As Integer, X As Single, Y As Single)
  If (Button = 2) Then
    Dim nID As Long
    nID = PopupMenu1.ShowAtCursor
    If (nID > 0) Then
      MsgBox "You have selected the '" & PopupMenu1.Command(nID).Caption & "'"
    End If
  End If
End Sub
```

property PopupMenu.Debug as Boolean

Retrieves or sets a value that indicating whether the item's identifier is visible.

Type	Description
Boolean	A Boolean expression that specifies whether the control displays the identifiers for the items.

By default, the Debug property is False. If the Debug property is set on True, the control shows the identifiers of the visible items. The [ID](#) property indicates the identifier of the item, and it is the same as the ID parameter of the [Add](#) method. The following sample shows the identifiers of the items (between [] brackets), while the popup menu control is shown:



method PopupMenu.Detach ()

Detaches the window's menu, attached by the Attach method.

Type	Description
	Detaches the menu assigned with the Attach method. Use the Command event to notify your application that the user clicks an item.

property PopupMenu.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.

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

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

property PopupMenu.FlatImageWidth as Long

Specifies the width of the column to display the icons/images when the control's MenuAppearance is exMenuFlat.

Type	Description
Long	A Long expression that specifies the width of the column that displays icons/images/check or radio buttons, when the control's Appearance is Flat.

By default, the FlatImageWidth property is 16 pixels wide. Use the FlatImageWidth property to specify the width of the column that displays icons/images/check or radio buttons. The Image / HTMLImage property assigns an icon / picture to the item. The tag can be used in the Caption property of the Item object to display an Icon or a custom-size picture.

property PopupMenu.Font as IFontDisp

Retrieves or sets the control's font.

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

Use the Font property to assign a font to the control.

property PopupMenu.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 change the control's foreground color. Use the [BackColor](#) property to change the control's background color.

property PopupMenu.HAlign as HAlignEnum

Specify how the control positions the shortcut menu horizontally.

Type	Description
HAlignEnum	A HAlignEnum expression that indicates how the control shows horizontally the popup menu.

The [VAlign](#) and HAlign properties specify how the control positions the shortcut menu relative to the cursor position.

method PopupMenu.Hide ()

Hides the control.

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

Use the Hide method to closes the popup control. Use the [CloseOnClick](#) property to specify how the control is closed.

property PopupMenu.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). Use the [Images](#) method to assign a collection of icons to the control. Use the [Image](#) property to assign a single icon to an item. Use the HTMLPicture property to display custom sized pictures in the menu control.

property PopupMenu.hWnd as Long

Retrieves the control window's handle.

Type	Description
Long	A long that identify a window's handle that hosts the popup menu.

Some containers might use this property.

method PopupMenu.Images (Handle as Variant)

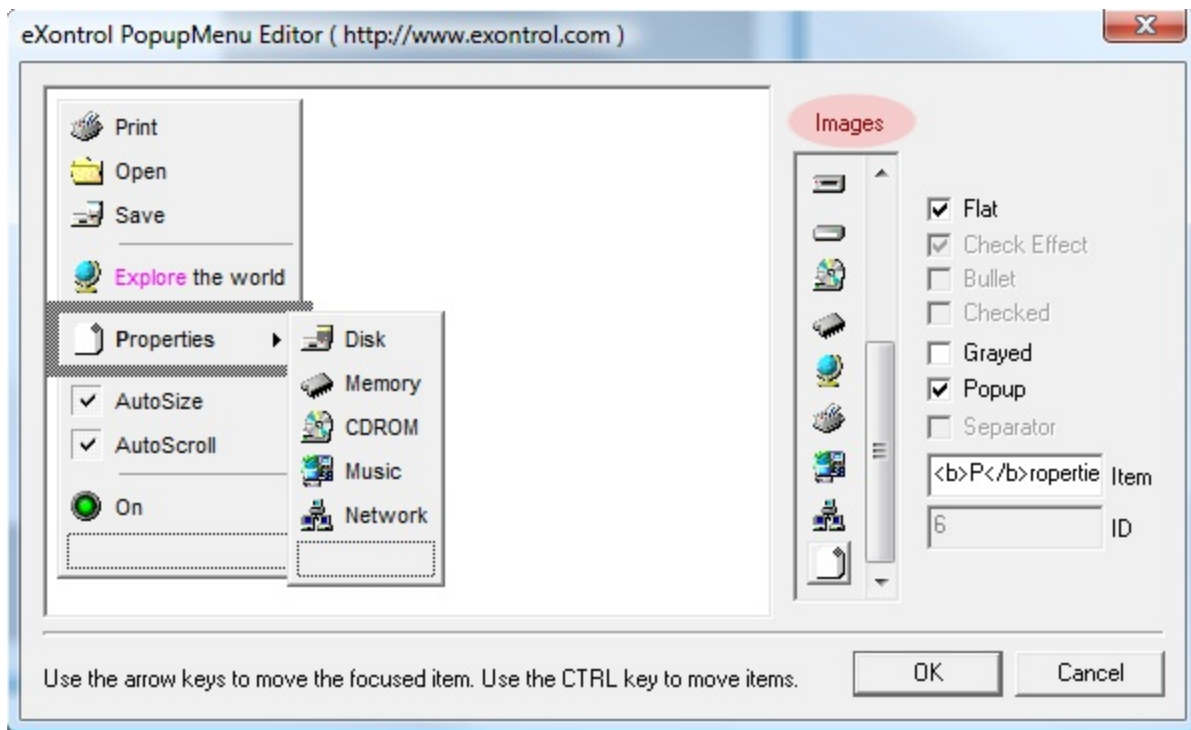
Sets the control's image list at runtime.

Type	Description
Handle as Variant	<p>The Handle parameter can be:</p> <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>)• 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>)• 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>)• 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>)• 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 Images method may be used to add a list of icons to your menu. Use the [HTMLPicture](#) property to display custom sized pictures in the menu. Use the [Image](#) property to specify the index of the single icon being displays in the item. Use the HTML tag to display multiple icons in the item anywhere in the item's caption.

You can add your icons at design mode, by dragging the icon files to images panel as shown bellow.



The following sample uses the Microsoft ImageList Control to set the control's image list at runtime:

```
PopupMenu1.Images ImageList1.hImageList
```


property PopupMenu.Item (ID as Variant) as Item

Returns a specific Item object given its caption or its identifier.

Type	Description
ID as Variant	A long expression that indicates the identifier of the item being accessed, or a string expression that specifies the caption of the item.
Item	An Item object being retrieved.

Use the Item property to access an Item given its identifier or its caption. The Item property looks recursively for an item that matches your identifier of specified caption. The [ID](#) property specifies the identifier of the item. The [Caption](#) property specifies the item's HTML caption. Use the [ItemByData](#) property to search for an item giving its associated data ([ItemData](#) property)

property PopupMenu.ItemByData (ItemData as Variant) as Item

Searches for a specific Item object given its associated data.

Type	Description
ItemData as Variant	A VARIANT expression looked for.
Item	An Item object being retrieved.

The ItemByData property retrieves the Item object with the associated data. Use the [ItemData](#) property to associate any extra data to your item. Use the [Visible](#) property to hide or show a specified item. Use the [Item](#) property to access an Item given its identifier or its caption.

property PopupMenu.ItemHeight as Long

Specifies the item's height in pixels.

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

Use the ItemHeight property to change the item's height. By default, the ItemHeight property is 20 pixels.

property PopupMenu.Items as Menu

Retrieves a Menu object to handle adding, removing or changing items at runtime.

Type	Description
Menu	A Menu object that contains a collection of Item objects.

Use the Items property to access to the control's items. Use the [Add](#) property to add a new item to the menu.

The following sample shows hot to add a new separator:

```
PopupMenu1.Items.Add "", Separator
```

property PopupMenu.SelBackColor as Color

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

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

Use the SelBackColor and [SelfForeColor](#) properties to specify the background and foreground colors for the selected item.

property PopupMenu.SelForeColor as Color

Retrieves or sets a color that indicates the selection's text color.

Type	Description
Color	A color expression that indicates the selection's text color.

Use the [SelBackColor](#) and SelForeColor properties to specify the background and foreground colors for the selected item.

property PopupMenu.ShadowColor as Color

Specifies the shadow color.

Type	Description
Color	A color expression that indicates the shadow color.

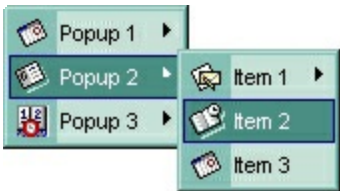
The shadow color is used to paint the left side of the menu when the menu's style is Flat.

method PopupMenu.Show (X as OLE_XPOS_PIXELS, Y as OLE_YPOS_PIXELS)

Displays the popup menu at given coordinates, and retrieves the menu command identifier.

Type	Description
X as OLE_XPOS_PIXELS	A long expression that indicates the horizontal position in pixels that should be relative to the screen.
Y as OLE_YPOS_PIXELS	A long expression that indicates the vertical position in pixels that should be relative to the screen.
Return	Description
Long	A long expression that indicates the identifier of the selected item.

Use the Show property to display the control at given coordinates. Use the [ShowAtCursor](#) property to display the shortcut menu at cursor. Use the [ShowAtWindow](#) property to show the popup menu control relative to specified window. The [Command](#) event is never fired during the Show or ShowAtCursor property.



The following sample shows how to use the Show method:

```
Private Type POINTAPI
    x As Long
    y As Long
End Type

Private Declare Function ClientToScreen Lib "user32" (ByVal hwnd As Long, lpPoint As POINTAPI) As Long

Private Sub Form_MouseUp(Button As Integer, Shift As Integer, x As Single, y As Single)
    If (Button = 2) Then
        Dim nID As Long
        Dim p As POINTAPI
        p.x = x / Screen.TwipsPerPixelX
        p.y = y / Screen.TwipsPerPixelY
        ClientToScreen Me.hwnd, p
```



```
nID = PopupMenu1.Show(p.x, p.y)
```

```
If (nID = 1234) Then
```

```
    PopupMenu1.Command(1234).Image = (PopupMenu1.Command(1234).Image +  
1) Mod 2
```

```
End If
```

```
End If
```

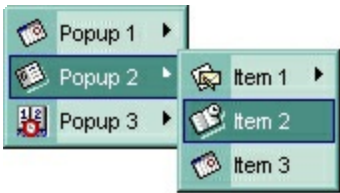
```
End Sub
```

property PopupMenu.ShowAtCursor as Long

Shows the popup menu at cursor position

Type	Description
Long	A long expression that indicates the identifier of the item selected.

The ShowAtCursor property displays the control at cursor coordinates. Use the [Show](#) property if you need to specify the coordinates where the popup menu should be displayed. Use the [ShowAtWindow](#) property to show the popup menu control relative to specified window.



The following sample shows how to display the popup at cursor coordinates:

```
Private Sub Form_MouseUp(Button As Integer, Shift As Integer, X As Single, Y As Single)
  If (Button = 2) Then
    Dim nID As Long
    nID = PopupMenu1.ShowAtCursor
    If (nID = 1234) Then
      PopupMenu1.Command(1234).Image = (PopupMenu1.Command(1234).Image +
1) Mod 2
    End If
  End If
End Sub
```

method PopupMenu.ShowAtWindow (hWnd as Long, Align as MenuAlignEnum, [AlignClientRect as Variant])

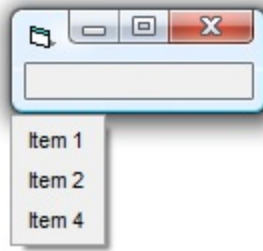
Displays the popup menu relative to the given window.

Type	Description
hWnd as Long	A Long expression that specifies the handle of the window that specifies the relative position of the popup menu control. If the hWnd parameter is 0, the popup is aligned relative to the container window.
Align as MenuAlignEnum	A MenuAlignEnum expression that specifies the alignment of the popup menu relative to the hWnd window.
AlignClientRect as Variant	A String expression that defines the client rectangle to align the popup control relative to the specified window by hWnd parameter. If missing, the popup is aligned to hWnd window's area. If AlignClientRect parameter is specified, the popup control is aligned to the specified client area relative to hWnd. The string format is as <i>"left,top,width,height"</i> , where left indicates the left coordinate, the top indicates the top coordinate, and so on. For instance, "10,10,16,16" indicates a rectangle of 16x16 size that's positioned at 10x10 point relative to the window. If the C: is present in front, the coordinates are relative to window's client area as for instance: "C:10,10,16,16"
Return	Description
Long	A long expression that specifies the identifier of the item being selected.

Use the ShowAtWindow property to show the popup menu relative to the given window. Use the [Show](#) method to show the control at specified coordinates. Use the [ShowAtCursor](#) property to show the popup menu at cursor position. If the hWnd parameter is 0, the popup menu is aligned to container window. The AlignClientRect parameter defines the inside client area to be used to align the popup control.

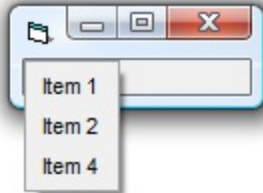
The following VB sample aligns the popup control to the bottom-left corner of the form:

```
Private Sub Form_Click()  
    With Me  
        Debug.Print PopupMenu1.ShowAtWindow(hWnd, MenuBottom)  
    End With  
End Sub
```



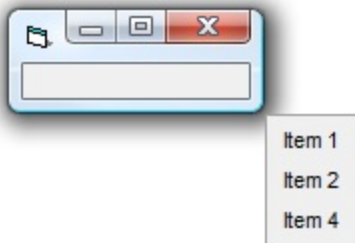
The following VB sample aligns the popup control to the top-left corner of the form's client area:

```
Private Sub Form_Click()  
    With Me  
        Debug.Print PopupMenu1.ShowAtWindow(hWnd, MenuBottom, "C:0,0")  
    End With  
End Sub
```



The following VB sample aligns the popup control to the bottom-right corner of the form:

```
Private Sub Form_Click()  
    With Me  
        Debug.Print PopupMenu1.ShowAtWindow(hWnd, MenuBottom, .Width /  
Screen.TwipsPerPixelX & "," & .Height / Screen.TwipsPerPixelY)  
    End With  
End Sub
```



property PopupMenu.VAlign as VAlignEnum

Specify how the control positions the shortcut menu vertically.

Type	Description
VAlignEnum	A VAlignEnum expression that specify how the control positions the shortcut menu vertically.

The VAlign and [HAlign](#) properties specify how the control positions the shortcut menu relative to the cursor position.

property PopupMenu.Version as String

Retrieves the control's version.

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

The Version property gets the control's version.

MenuButton 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: {462D5053-2D60-4022-B583-7E34AA0F90B7}. The object's program identifier is: "Exontrol.ExPopupMenu". The /COM object module is: "ExPMenu.dll"

The MenuButton object supports the following events:

Name	Description
Click	Occurs when the user clicks the button.
Dismiss	Occurs when the menu is closed.
Push	Occurs when the menu is opened.

event Click ()

Occurs when the user clicks the button.

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

The Click event is fired when the user clicks the button.

The following sample changes the button's caption when user selects a new item:

```
Private Sub MenuButton1_Click()  
    With MenuButton1  
        .Caption = PopupMenu1(.LastID).Caption  
    End With  
End Sub
```


event Dismiss ()

Occurs when the menu is closed.

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

The Dismiss event is fired when the assigned menu is closed.

event Push ()

Occurs when the menu is opened

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

The Push event is fired when the user clicks the drop down button.

PopupMenu 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: {462D5053-2D60-4022-B583-7E34AA0F90B7}. The object's program identifier is: "Exontrol.ExPopupMenu". The /COM object module is: "ExPMenu.dll"

The PopupMenu object supports the following events:

Name	Description
Command	Occurs when the user selects a menu bar item..
MouseMove	Occurs when the user moves the mouse.

event Command (ID as Long)

Occurs when the user selects a menu bar item.

Type	Description
ID as Long	A long expression that indicates the identifier of the command being selected.

Use the Command event to notify your application when an item is selected. The Command event is fired only if the menu is attached to a form using the [Attach](#) method. The Command event is never fired if you display the control using the [Show](#) or [ShowAtCursor](#) properties. Use the Attach method to attach a menu bar to a dialog or a form. The Command event is fired when user clicks or selects an item from the dialog/form's menu bar

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

C#private void Command(object sender,int ID)
{
}

VBPrivate Sub Command(ByVal sender As System.Object,ByVal ID As Integer) Handles
Command
End Sub

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

C#private void Command(object sender,
AxEXPOPUPMENULib._IPopupMenuEvents_CommandEvent e)
{
}

C++void OnCommand(long ID)
{
}

C++ Buildervoid __fastcall Command(TObject *Sender,long ID)
{
}

Delphiprocedure Command(ASender: TObject; ID : Integer);

```
begin  
end;
```

Delphi 8
(.NET
only)

```
procedure Command(sender: System.Object; e:  
AxEXPOPUPMENULib._IPopupMenuEvents_CommandEvent);  
begin  
end;
```

Power...

```
begin event Command(long ID)  
end event Command
```

VB.NET

```
Private Sub Command(ByVal sender As System.Object, ByVal e As  
AxEXPOPUPMENULib._IPopupMenuEvents_CommandEvent) Handles Command  
End Sub
```

VB6

```
Private Sub Command(ByVal ID As Long)  
End Sub
```

VBA

```
Private Sub Command(ByVal ID As Long)  
End Sub
```

VFP

```
LPARAMETERS ID
```

Xbas...

```
PROCEDURE OnCommand(oPopupMenu,ID)  
RETURN
```

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

Java...

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

VBSc...

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

Visual
Data...

```
Procedure OnComCommand Integer IID
    Forward Send OnComCommand IID
End_Procedure
```

Visual
Objects

```
METHOD OCX_Command(ID) CLASS MainDialog
RETURN NIL
```

X++

```
void onEvent_Command(int _ID)
{
}
```

XBasic

```
function Command as v (ID as N)
end function
```

dBASE

```
function nativeObject_Command(ID)
return
```

event MouseEventArgs (Button as Integer, Shift as Integer, X as OLE_XPOS_PIXELS, Y as OLE_YPOS_PIXELS, CursorInside as Boolean)

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
CursorInside as Boolean	Specifies whether the cursor is inside of the popup menu.

The MouseEventArgs event is generated continually as the mouse pointer moves across objects.

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

C#

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

VB

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

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

C#

```
private void MouseEventArgs(object sender,
AxEXPOPUPMENULib._IPopupMenuEvents_MouseMoveEvent e)
{
}
```

C++

```
void OnMouseMove(short Button,short Shift,long X,long Y,BOOL CursorInside)
{
}
```

**C++
Builder**

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

Delphi

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

**Delphi 8
(.NET
only)**

```
procedure MouseMoveEvent(sender: System.Object; e:
AxEXPOPUPMENULib._IPopupMenuEvents_MouseMoveEvent);
begin
end;
```

Powe...

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

VB.NET

```
Private Sub MouseMoveEvent(ByVal sender As System.Object, ByVal e As
AxEXPOPUPMENULib._IPopupMenuEvents_MouseMoveEvent) Handles
MouseMoveEvent
End Sub
```

VB6

```
Private Sub MouseMove(Button As Integer,Shift As Integer,X As Single,Y As
Single,CursorInside As Boolean)
End Sub
```

VBA

```
Private Sub MouseMove(ByVal Button As Integer,ByVal Shift As Integer,ByVal X As
Long,ByVal Y As Long,ByVal CursorInside As Boolean)
End Sub
```


VFP

LPARAMETERS Button,Shift,X,Y,CursorInside

Xbas... PROCEDURE OnMouseMove(oPopupMenu,Button,Shift,X,Y,CursorInside)
RETURN

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

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

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

Visual Data... Procedure OnComMouseMove Short IButton Short IShift OLE_XPOS_PIXELS IIX
OLE_YPOS_PIXELS IY Boolean IICursorInside
Forward Send OnComMouseMove IButton IShift IIX IY IICursorInside
End_Procedure

Visual Objects METHOD OCX_MouseMove(Button,Shift,X,Y,CursorInside) CLASS MainDialog
RETURN NIL

X++ void onEvent_MouseMove(int _Button,int _Shift,int _X,int _Y,boolean
_CursorInside)
{
}

XBasic function MouseMove as v (Button as N,Shift as N,X as
OLE::Exontrol.ExPopupMenu.1::OLE_XPOS_PIXELS,Y as
OLE::Exontrol.ExPopupMenu.1::OLE_YPOS_PIXELS,CursorInside as L)
end function

dBASE function nativeObject_MouseMove(Button,Shift,X,Y,CursorInside)

