

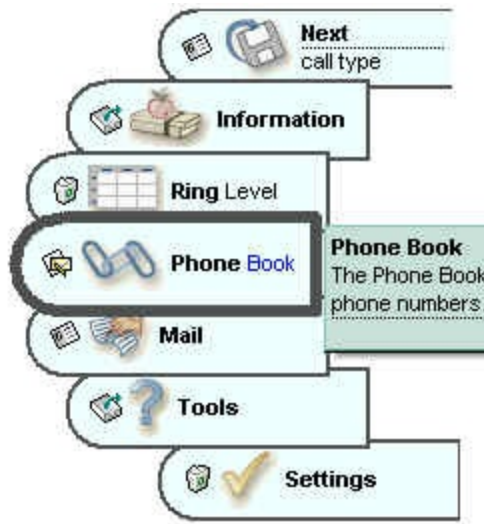


ExRoList

Exontrol's new exRoList control handles and displays lists on an elliptic shape. The exRoList rotates the list to let items being visible, and so no scrolls are required. The exRoList 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:

- **Skinnable Interface** support (ability to apply a skin to the any background part)
- **WYSWYG Template/Layout** editor
- unlimited options to display items using **HTML** format
- ability to assign multiple HTML lines **tooltips** to items
- ability to assign icons, pictures to items
- ability to use BASE64 encoded icons or pictures.
- incremental search feature
- nice animation when the list is rotated
- mouse wheel support
- keyboard support



Ž ExRoList 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

Aligns an item. Use the [Alignment](#) property to align an item.

Name	Value	Description
exLeft	0	Aligns the caption to the left side.
exCenter	1	Centers the caption.
exRight	2	Aligns the caption to the right side.

constants AppearanceEnum

Specifies the source's appearance.

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

constants BackgroundPartEnum

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

Name	Value	Description
exSelectNode	0	Specifies the visual appearance for the selected node.
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 PictureBoxDisplayEnum

Specifies how a picture object is displayed.

Name	Value	Description
UpperLeft	0	UpperLeft
UpperCenter	1	UpperCenter
UpperRight	2	UpperRight
MiddleLeft	16	MiddleLeft
MiddleCenter	17	MiddleCenter
MiddleRight	18	MiddleRight
LowerLeft	32	LowerLeft
LowerCenter	33	LowerCenter
LowerRight	34	LowerRight
Tile	48	Tile
Stretch	49	Stretch

Appearance object

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

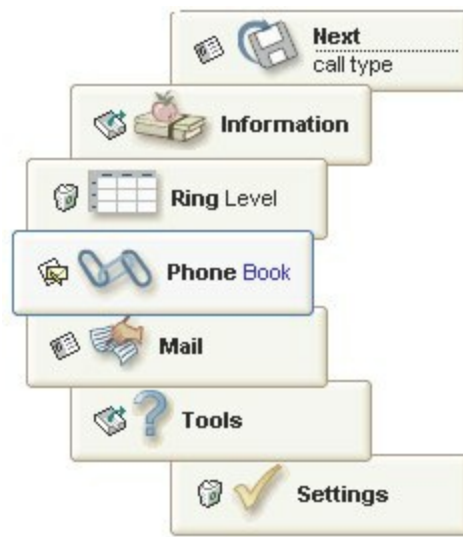
Name	Description
Add	Adds or replaces a skin object to the control.
Clear	Removes all skins in the control.
Remove	Removes a specific skin from the control.
RenderType	Specifies the way colored EBN objects are displayed on the component.

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

Adds or replaces a skin object to the control.

Type	Description
ID as Long	A Long expression that indicates the index of the skin being added or replaced. The value must be between 1 and 126, so Appearance collection should holds no more than 126 elements.
Skin as Variant	<p>A string expression that indicates the path to the skin file (*.ebn), or a string expression that indicates 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. The Exontrol's exButton component installs a skin builder that can be used to create new skins.</p> <p>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)).</p>
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.



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:

- border, [Appearance](#) property
- selected item, tooltips, [Background](#) property
- item, [BackColor](#) property
- items, [ItemBackColor](#) property

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

With Rollist1

With .VisualAppearance

.Add &H22, App.Path + "\\select.ebn"

End With

.SelBackColor = RGB(0,0,255) Or &H22000000

End With

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

```
#include "Appearance.h"
m_xmlgrid.GetVisualAppearance().Add( 0x22,
COleVariant(_T("D:\\Temp\\ExRollist.Help\\select.ebn")) );
m_xmlgrid.SetSelBackColor( RGB(0,0,255) | 0x22000000 );
```

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

```
With AxRollist1
    With .VisualAppearance
        .Add(&H22, "D:\\Temp\\ExRollist.Help\\select.ebn")
    End With
    .Template = "SelBackColor = 587137024"
End With
```

where the 587137024 value is the hexa representation of 0x22FF0000

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

```
axRollist1.VisualAppearance.Add(0x22, "d:\\temp\\ExRollist.Help\\select.ebn");
axRollist1.Template = "SelBackColor = 587137024";
```

where the 587137024 value is the hexa representation of 0x22FF0000.

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

```
With thisform.Rollist1
    With .VisualAppearance
        .Add(34, "D:\\Temp\\ExRollist.Help\\select.ebn")
    EndWith
    .SelBackColor = RGB(0,0,255) + 570425344
EndWith
```

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

BeginUpdate

VisualAppearance

{

Add(1,

"gBFLBCJwBAEHhEJAEGg4BcoDg6AABACAxWgKBADQKAAYDIKsEQGGIZRhhGlwAgaFIXQK
)

Add(2,

"gBFLBCJwBAEHhEJAEGg4BNYCg6AABACAxWgKBADQKAAYDIKsEQGGIZRhhGlwAgaFIXQK

}

Background(0) = 33554432

' ItemBackColor = RGB(240,255,255)

ItemBackColor = 16777216

BackColor = RGB(255,255,255)

BorderColor = RGB(80,80,80)

ItemWidth = 0 ' Lets the control calculates the item's width based on its content

Images("gBJJgBggAAwAAgACEKAD/hz/EMNh8TIRNGwAjEZAEXjAojJAjlgjIBAEijUlK8pIUrlktl

Images("gBJJgBggAAwAAgACEKAD/hz/EMNh8TIRNGwAjEZAEXjAojJAjlgjIBAEijUlK8pIUrlktl

Images("gBJJgBggAAkGAAQhIAf8Nf4hhkOiRCJo2AEXjAAi0XFEYIEYhUXAIAEEZi8hk0pIUrlktl

Images("gBJJgBggAAwAAgACEKAD/hz/EMNh8TIRNGwAjEZAEXjAojJAjlgjIBAEijUlK8pIUrlktl

Images("gBJJgBggAAwAAgACEKAD/hz/EMNh8TIRNGwAjEZAEXjAojJAjlgjIBAEijUlK8pIUrlktl

Images("gBJJgBggAAwAAgACEKAD/hz/EMNh8TIRNGwAjEZAEXjAojJAjlgjIBAEijUlK8pIUrlktl

Add("**Phone Book**", 1)

{

ToolTip = "The Phone Book helps you to organize phone numbers.

exRollList"

ToolTipTitle = "Phone Book"

Picture =

"gBHJJGHA5MltAAjAD3AAAAAsPK50SphP0TPyfNaIT52RCoOySVCAkCATkhTiyRUUnIK7SimWL

}

Add("**Mail**", 2)

{

ToolTip = "The Phone Book helps you to organize phone numbers.

exRollList"

ToolTipTitle = "Mail"

Picture =

"gBHJJGHA5MIjAEIe4AAAFhwwLR6HRQN5FMp6iyFikaN6PLUXLRvQshjx6kqaMpFHUXR8XL

}

Add("**Tools**", 3)

{

ToolTip = "For almost all that time NAG has employed sophisticated software tools to ensure that its software maintains a high standard and is as widely available as possible.

exRollList"

ToolTipTitle = "Tools"

Picture =

"gBHJJGHA5MIYAAjAD3AAAAAsPLyBUpeRaINEVi6vNCUV5yRcdjhySi4QqekinYSPTYvR6nXEtI

}

Add("**Settings**", 4)

{

ToolTip = "Tribes, Clans, Kiths, and Traditions, based in the World of Darkness, built on by the individuals in the chronicle. But what is that World of Darkness without the background unique to the Camarilla chronicle.

exRollList"

ToolTipTitle = "Settings"

Picture =

```
"gBHJJGHA5MIgAAjAD3AAAAsPU6ZMMROixiUWiQjNaxUZ9WKsUS7jUiOkiPq7VklVknViHX
```

```
}
```

```
Add("Access", 5)
```

```
{
```

ToolTip = "Work smarter with Access version 2002, the Office XP database management program. Whether you're a novice database user or an experienced programmer, you can now build powerful, customizable solutions that integrate easily with the Web and enterprise data sources.

exRolList"

```
ToolTipTitle = "Access"
```

```
Picture =
```

```
"gBHJJGHA5MIjAEIe4AAAFhx+R6ZU5dLCnM5hU5xMKxjUcNaxPcfPZ0WKfKisUS7k0qPq7R8
```

```
}
```

```
Add("Networks", 6)
```

```
{
```

ToolTip = "Nortel Networks is transforming how the world communicates and exchanges information.

exRolList"

```
ToolTipTitle = "Networks"
```

```
Picture =
```

```
"gBHJJGHA5MIkAAjAD3AAAAsPMiOUZuRyuPKYiyYXCDjCDUa4iSuTCtXEjYatUaZVsoVrCZq4
```

```
}
```

```
Add("Divert", 1)
```

```
{
```

ToolTip = "Divert sockets enable both IP packet interception and injection on the end-hosts as well as on routers.

exRolList"

```
ToolTipTitle = "Divert"
```

```
Picture =
```

```
"gBHJJGHA5MIkAAjAD3AAAAsPEpFOAIKBwHZbPY7Mp7IsYlPlQsVQpbOCQjCQjcoQqbOErF
```

```
}
```

```
Add("Next
```

```
call type", 2)
```

```
{
    ToolTip = "Before answering calls or any other call processing, call discrimination
prevents specific DNIS.
exRolList"
    ToolTipTitle = "Next Call Type"
    Picture =
"gBHJJGHA5MloAAjAD3AAAAsPGBQOA7iQ7LZ7IplQsSSBbj0eMsXTZIkkaUpwlBwPaFOCQU

}
Add("Information", 3)
{
    ToolTip = "Please is the world's largest free reference site. Here you can find facts on
thousands of subjects including sports.
exRolList"
    ToolTipTitle = "Information"
    Picture =
"gBHJJGHA5MIsAAjAD3AAAAsPJhBFZWJY0NRVGh1L5HOPpjhpLR/OZHkUhQ53RJ9MkpNq.

}
Add("Ring Level", 4)
{
    ToolTip = "Enter 440499 as a code. A monster will roar to confirm correct code entry.
exRolList"
    ToolTipTitle = "Ring Level"
    Picture =
"gBHJJGHA5MlpAAjAD3AAAAsPDZQPawWCIXcUXcXjK7YsZYsdaDQXbbaDFbckcsolLwlbwbk

}
EndUpdate
Refresh
```

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:

- border, [Appearance](#) property
- selected item, tooltips, [Background](#) property
- item, [BackColor](#) property
- items, [ItemBackColor](#) 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:

- border, [Appearance](#) property
- selected item, tooltips, [Background](#) property
- item, [BackColor](#) property
- items, [ItemBackColor](#) property


property Appearance.RenderType as Long

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

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

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

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

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

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

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

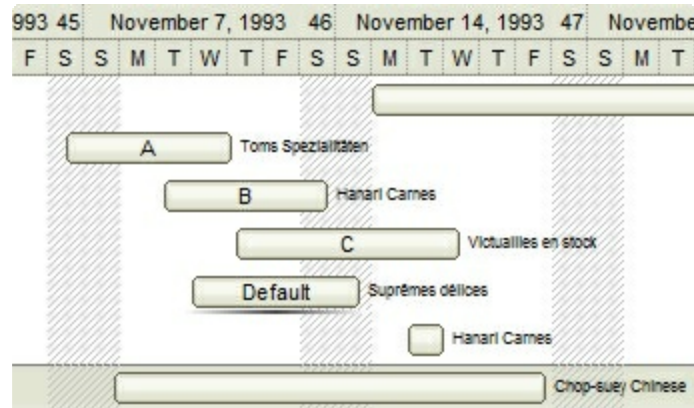
In the following screen shot the following objects displays the current EBN with a different color:

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

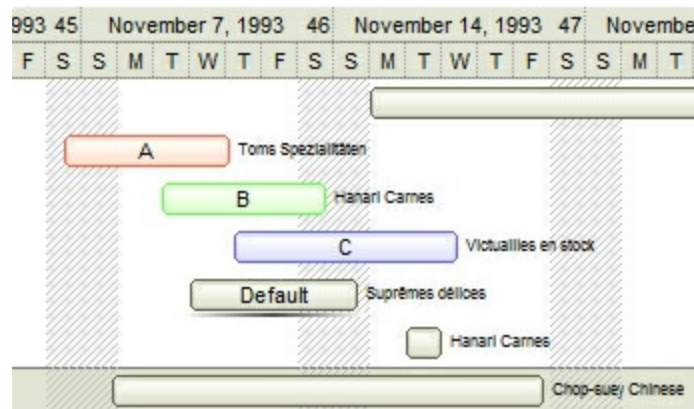
- "C" in Blue (RGB(0,0,255) , for instance the bar's property exBarColor is 0x1FF0000
- "Default", no color is specified, for instance the bar's property exBarColor is 0x1000000

The RenderType property could be one of the following:

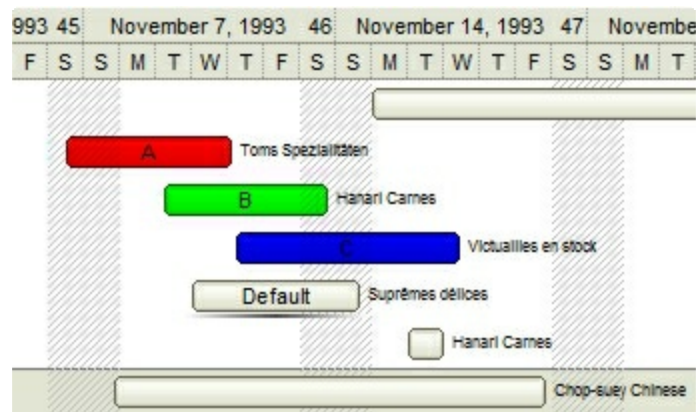
- **-3, no color is applied.** For instance, the BackColorHeader = &H1FF0000 is displayed as would be .BackColorHeader = &H1000000, so the 0xFF0000 color (Blue color) is ignored. You can use this option to allow the control displays the EBN colors or not.



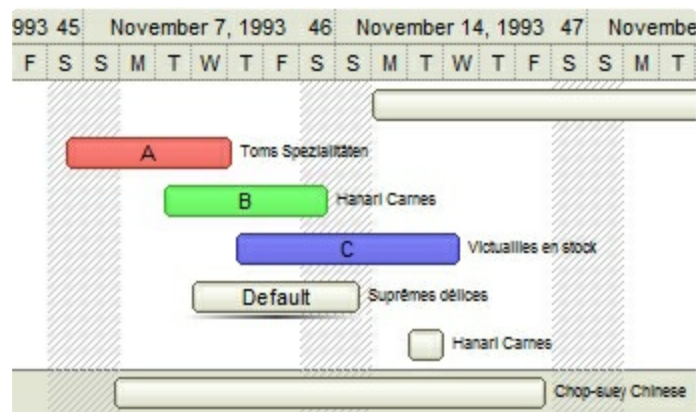
- **-2, OR-color scheme.** The color to be applied on the part of the control is a OR bit combination between the original EBN color and the specified color. For instance, the BackColorHeader = &H1FF0000, applies the OR bit for the entire Blue channel, or in other words, it applies a less Blue to the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



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

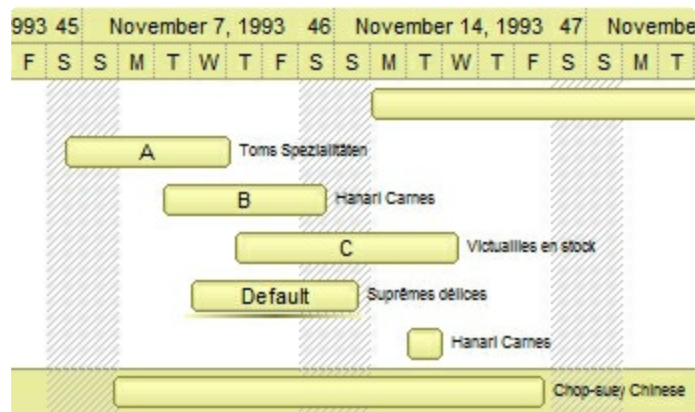


- **0, default**, the specified color is applied to the EBN. For instance, the BackColorHeader = &H1FF0000, applies a Blue color to the object. This option could be used to specify any color for the part of the components, that support EBN objects, not only solid colors.

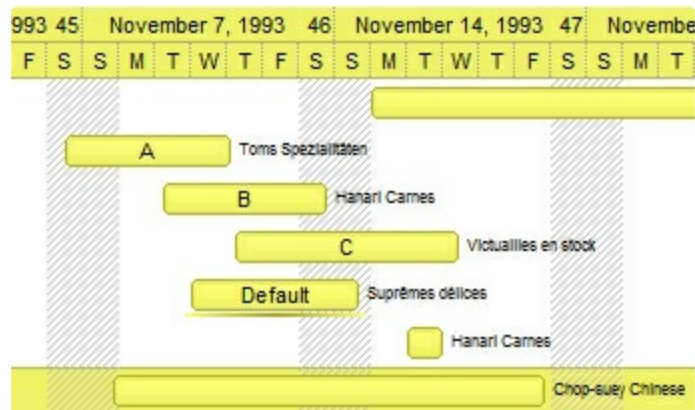


- **0xAABBGGRR**, where the AA a value between 0 to 255, which indicates the transparency, and RR, GG, BB the red, green and blue values. This option applies the same color to all parts that displays EBN objects, whit ignoring any specified color in the color property. For instance, the RenderType on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

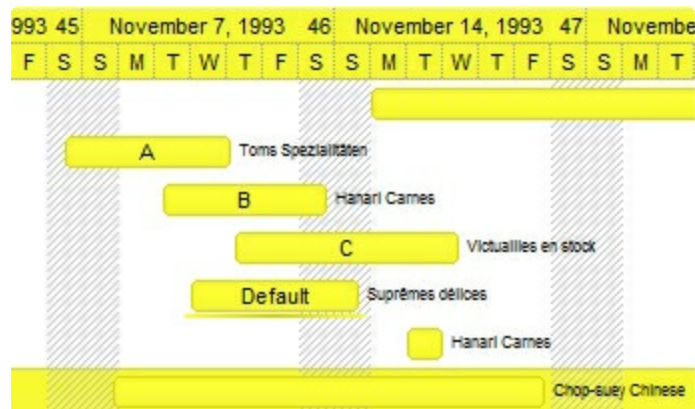
The following picture shows the control with the RenderType property on 0x4000FFFF (25% Yellow, 0x40 or 64 in decimal is 25% from 256):



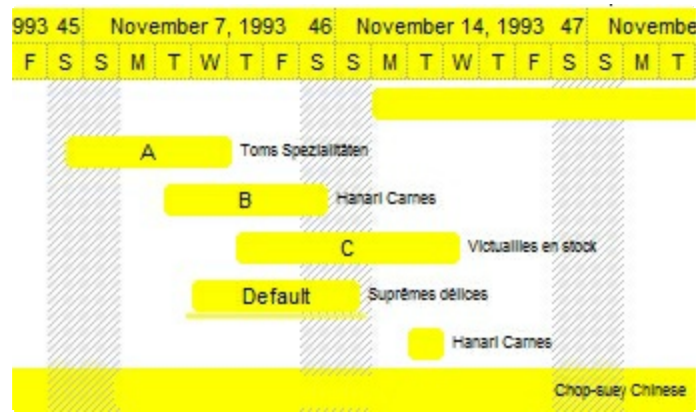
The following picture shows the control with the *RenderType* property on `0x800FFFFF` (50% Yellow, `0x80` or 128 in decimal is 50% from 256):



The following picture shows the control with the *RenderType* property on `0xC00FFFFF` (75% Yellow, `0xC0` or 192 in decimal is 75% from 256):



The following picture shows the control with the *RenderType* property on `0xFF00FFFF` (100% Yellow, `0xFF` or 255 in decimal is 100% from 255):



Item object

The Item object holds information about an item. An item can have associated an icon and multiple lines HTML text. The Item object supports the following methods and properties:

Name	Description
Alignment	Specifies the item's alignment.
BackColor	Specifies the item's background color.
Bold	Sets or returns whether or not the item should appear in bold.
Caption	Returns or sets a value that indicates the caption of the item.
ClearBackColor	Clears the item's background color.
ClearForeColor	Clears the item's foreground color.
ForeColor	Specifies the item's foreground color.
Image	Specifies a value that indicates the index of icon being displayed.
Italic	Sets or returns whether or not the item should appear in italic.
Picture	Specifies the item's picture.
StrikeOut	Sets or returns whether or not the item should appear in strikeouts.
ToolTip	Specifies the description for item's tooltip.
ToolTipTitle	Specifies the title of the item's tooltip.
Underline	Sets or returns whether or not the item should appear in underline.
UserData	Specifies an extra value associated to the item.

property Item.Alignment as AlignmentEnum

Specifies the item's alignment.

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

Use the Alignment property to align the item's caption. The [Caption](#) property specifies the item's caption. By default, the caption of the item is left aligned.

property Item.BackColor as Color

Specifies 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 colors for an item. Use the [ItemBackColor](#) property to specify the item's default background color. Use the [ClearBackColor](#) method to clear the item's background color specified with the BackColor property.

property Item.Bold as Boolean

Sets or returns whether or not the item should appear in bold.

Type	Description
Boolean	A boolean expression that specifies whether or not the item should appear in bold.

Use the Bold property to bold an item. Use the HTML tag in [Caption](#) property to specify parts that should appear in bold. Use the [Italic](#) property to specify that item should appear in italic. Use the [StrikeOut](#) property to specify that item should appear in strikeout. Use the [Underline](#) property to specify that item should appear in underline.

property Item.Caption as String

Returns or sets a value that indicates the caption of the item.


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

Use the Caption property to specify the item's caption. Use the **** HTML tag to insert icons inside the item's caption.

The Caption property supports built-in HTML tags like follows:

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

<solidline> or <dotline>).

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

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

height of the font. For instance the "<out 000000>

<fgcolor=FFFFFF>outlined</fgcolor></out>" generates the following picture:



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



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



You can pass set the item's caption when adding the item using the [Add](#) method. Use the [Bold](#) property to specify whether or not the item should appear n bold. Use the [Image](#), [Picture](#) properties to assign icons or pictures to your item.

method Item.ClearBackColor ()

Clears the item's background color.

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

The ClearBackColor method has no effect if the [BackColor](#) property wasn't used. The item's background color is defined by the [BackColor](#) property, if the ClearBackColor method is called.

method Item.ClearForeColor ()

Clears the item's foreground color.

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

The ClearForeColor method has no effect if the [ForeColor](#) property wasn't called. The item's foreground color is defined by the [ForeColor](#) property, if the ClearForeColor method is called.

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 colors for an item. Use the [ForeColor](#) property to specify the item's default foreground color. Use the [ClearForeColor](#) method to clear the item's foreground color specified with the ForeColor property.

property Item.Image as Long

Specifies a value that indicates the index of icon being displayed.

Type	Description
Long	A long expression that indicates the index of icon being displayed.

Use the Image property to assign an icon to the item. By default, the Image property is 0. If the Image property is 0. the item has no icon associated. Use the [Add](#) method property to assign an icon to the item when it is created. Use the [Images](#) method to assign a list of icons to the control. Use the [Picture](#) property to assign a picture to your item. Use the `` HTML tag to insert icons inside the item's caption.

property Item.Italic as Boolean

Sets or returns whether or not the item should appear in italic.

Type	Description
Boolean	A boolean expression that indicates whether or not item should appear in italic.

Use the Italic property to specify that item should appear in italic. Use the <i> HTML tag in [Caption](#) property to specify parts that should appear in italic. Use the [Bold](#) property to specify that item should appear in bold. Use the [StrikeOut](#) property to specify that item should appear in strikeout. Use the [Underline](#) property to specify that item should appear in underline.

property Item.Picture as Variant

Specifies the item's picture.

Type	Description
Variant	A Picture object that indicates the cell's picture. (A Picture object implements IPicture interface), 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.

Use the Picture property to assign a picture to the item. Use the [Image](#) property to assign an icon to your item. You can display transparent pictures, if your picture is saved using a transparent color.

property Item.StrikeOut as Boolean

Sets or returns whether or not the item should appear in knockout.

Type	Description
Boolean	A boolean expression that indicates whether or not item should appear in knockout.

Use the StrikeOut property to specify that item should appear in knockout. Use the <s> HTML tag in [Caption](#) property to specify parts that should appear in knockout. Use the [Bold](#) property to specify that item should appear in bold. Use the [Italic](#) property to specify that item should appear in italic. Use the [Underline](#) property to specify that item should appear in underline.

property Item.ToolTip as String

Specifies the description for item's tooltip.


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

Use the ToolTip property to assign a tooltip to an item. By default, the ToolTip property is empty. If the ToolTip property is empty, the item has no tooltip associated. Use the [ToolTipTitle](#) property to specify the title of the item's tooltip. Use the [ToolTipDelay](#) property to specify the time in ms that passes before the ToolTip appears.

The ToolTip property supports built-in HTML tags like follows:

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

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

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

- **<out rrggbb;width> ... </out>** shows the text with outlined characters, where rr/gg/bb represents the red/green/blue values of the outline color, 808080 if missing as gray,

width indicates the size of the outline, 1 if missing. The text color or <fgcolor> defines the color to show the inside text. The HTML tag can be used to define the height of the font. For instance the "<out 000000>

<fgcolor=FFFFFF>outlined</fgcolor></out>" generates the following picture:



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



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



property Item.ToolTipTitle as String

Specifies the title of the item's tooltip.

Type	Description
String	A string expression that indicates the title of the item's tooltip.

The ToolTipTitle property defines the title for the item's tooltip. The Use the [ToolTip](#) property to define the item's tooltip.

property Item.Underline as Boolean

Sets or returns whether or not the item should appear in underline.

Type	Description
Boolean	A boolean expression that specifies whether or not item should appear in underline.

Use the Underline property to specify that item should appear in underline. Use the <u> HTML tag in [Caption](#) property to specify parts that should appear in underline. Use the [Bold](#) property to specify that item should appear in bold. Use the [Italic](#) property to specify that item should appear in italic. Use the [StrikeOut](#) property to specify that item should appear in strikeout.

property Item.UserData as Variant

Specifies an extra value associated to the item.

Type	Description
Variant	A VARIANT value that indicates an extra value associated to the item.

The UserData property assign an extra value to the item.

Rollist 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: {08EA793D-92E4-4DA3-A66D-06E33F1F3945}. The object's program identifier is: "Exontrol.Rollist". The /COM object module is: "ExRolList.dll"

The RolList control handles and displays lists on an elliptic shape. The RolList object supports the following methods and properties:

Name	Description
Add	Adds an Item object to the collection and returns a reference to the newly created object.
AnchorFromPoint	Retrieves the identifier of the anchor from point.
Appearance	Retrieves or sets the control's appearance.
AttachTemplate	Attaches a script to the current object, including the events, from a string, file, a safe array of bytes.
AutoSearch	Enables or disables the incremental search feature.
BackColor	Specifies the control's background color.
Background	Returns or sets a value that indicates the background color for parts in the control.
BeginUpdate	Maintains performance when items are added to the control one at a time. This method prevents the control from painting until the EndUpdate method is called.
BorderColor	Specifies a value that indicates the color of the border for each item.
BorderHeight	Sets or retrieves a value that indicates the border height of the control.
BorderWidth	Sets or retrieves a value that indicates the border width of the control.
Count	Returns the number of items in the control.
Enabled	Enables or disables the control.
EndUpdate	Resumes painting the control after painting is suspended by the BeginUpdate method.
EventParam	Retrieves or sets a value that indicates the current's event parameter.
ExecuteTemplate	Executes a template and returns the result.
Font	Retrieves or sets the control's font.

ForeColor	Specifies the control's foreground color.
FormatAnchor	Specifies the visual effect for anchor elements in HTML captions.
HTMLPicture	Adds or replaces a picture in HTML captions.
hWnd	Retrieves the control's window handle.
Images	Sets at runtime the control's image list. The Handle should be a handle to an Image List Control.
ImageSize	Retrieves or sets the size of icons the control displays..
Item	Returns a specific Item from the collection.
ItemBackColor	Specifies the default background color for all items.
ItemBorderSize	Specifies the size of the border for items.
ItemFromPoint	Retrieves the item from cursor.
ItemHeight	Specifies the height of the item in pixels. If it is 0 the control computes the required height.
ItemWidth	Specifies the width of the item in pixels. If it is 0 the control computes the required width.
Picture	Retrieves or sets a graphic to be displayed in the control.
PictureDisplay	Retrieves or sets a value that indicates the way how the graphic is displayed on the control's background
Refresh	Refreshes the control.
Remove	Removes a specific item from the collection.
RemoveAllItems	Removes all items in the collection.
Replacelcon	Adds a new icon, replaces an icon or clears the control's image list.
RollDown	Selects the next item by rotating the list.
RollUp	Selects the previously item by rotating the list.
RoundCorner	Specifies a value that indicates whether the items have round corners.
Select	Selects a new item.
SelectBorderSize	Specifies a value that indicates the size of the border for selected item.
ShowImageList	Specifies whether the control's image list window is visible or hidden.
ShowToolTip	Shows the specified tooltip at given position.

[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.

[UseHandCursor](#) Specifies a value that indicates whether the control displays the hand cursor when cursor is over the items.

[Version](#) Retrieves the control's version.

[VisibleItems](#) Specifies a value that indicates the count of visible items.

[VisualAppearance](#) Retrieves the control's appearance.

Method Rollist.Add (Caption as String, [Image as Variant])

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

Type	Description
Caption as String	A string expression that defines the item's caption. Built-in HTML tags supported. Check the Caption property for all possible HTML tags.
Image as Variant	A long expression that specifies the index of icon being assigned to the item.
Return	Description
Item	An Item object being created.

Use the Add method to add new items to the control. Use the Caption property to retrieve the item's caption. Use the [Remove](#) method to remove a specific item. Use the [Refresh](#) method to refresh the control. Use the [Item](#) property to access to the items of the control. Use the [ToolTip](#) property to assign a tooltip to an item.

The following sample adds a new item:

```
With Rollist1
    .BeginUpdate
        With .Add("<b>New</b> Item", 1)
            .ToolTip = "Adds a new node"
        End With
    .EndUpdate
End With
```

property Rollist.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 Rollist1_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
    With Rollist1
        .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 AxRollist1_MouseMoveEvent(ByVal sender As System.Object, ByVal e As AxEXROLLISTLib.IRollistEvents_MouseMoveEvent) Handles AxRollist1.MouseMoveEvent
    With AxRollist1
        .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 axRollist1_MouseMoveEvent(object sender,
AxEXROLLISTLib._IRollistEvents_MouseMoveEvent e)
{
    axRollist1.ShowToolTip(axRollist1.get_AnchorFromPoint(-1, -1));
}
```

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

```
void OnMouseMoveRollist1(short Button, short Shift, long X, long Y)
{
    COleVariant vtEmpty; V_VT( &vtEmpty ) = VT_ERROR;
    m_rollist.ShowToolTip( m_rollist.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 .Rollist1
        .ShowToolTip(.AnchorFromPoint(-1, -1))
    EndWith
endwith
```

property Rollist.Appearance as AppearanceEnum

Retrieves or sets the control's appearance.

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

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



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

```
With Rollist1
    .BeginUpdate
        .VisualAppearance.Add &H16, "c:\temp\frame.ebn"
        .Appearance = &H16000000
        .BackColor = RGB(250, 250, 250)
    EndWith
```



```
.EndUpdate  
End With
```

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

```
With AxRollist1  
    .BeginUpdate()  
    .VisualAppearance.Add(&H16, "c:\temp\frame.ebn")  
    .Appearance = &H16000000  
    .BackColor = Color.FromArgb(250, 250, 250)  
    .EndUpdate()  
End With
```

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

```
axRollist1.BeginUpdate();  
axRollist1.VisualAppearance.Add(0x16, "c:\\temp\\frame.ebn");  
axRollist1.Appearance = (EXROLLISTLib.AppearanceEnum)0x16000000;  
axRollist1.BackColor = Color.FromArgb(250, 250, 250);  
axRollist1.EndUpdate();
```

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

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

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

```
with thisform.Rollist1  
    .BeginUpdate  
        .VisualAppearance.Add(0x16, "c:\temp\frame.ebn")  
        .Appearance = 0x16000000  
        .BackColor = RGB(250, 250, 250)  
    .EndUpdate  
endwith
```


method Rollist.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 Rollist1_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 Rollist.AutoSearch as Boolean

Enables or disables the incremental search feature.

Type	Description
Boolean	A boolean expression that indicates whether the incremental search feature is enabled or disabled.

Use the AutoSearch property to disable the incremental search feature. By default, the AutoSearch property is True. If the AutoSearch property is True, the control searches for an item as soon as user types characters.

property Rollist.BackColor as Color

Specifies the control's background color.

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

The BackColor property specifies the control's background color. Use the [ItemBackColor](#) property to define the default background color for items. Use the [ForeColor](#) property to specify the control's foreground color.

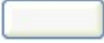
property Rollist.Background(Part as BackgroundPartEnum) as Color

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. The [Background](#)(exSelectNode) property indicates the selection background color. Shortly, we need to add a skin to the Appearance object using the Add method, and we need to set the last 7 bits in the Background property to indicates the index of the skin that we want to use. The sample applies the "" to the selected item:

```
With Rollist1
  With .VisualAppearance
    .Add &H22, App.Path + "\select.ebn"
  End With
  .SelBackColor = RGB(0,0,255) Or &H22000000
```

End With

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

```
#include "Appearance.h"
m_xmlgrid.GetVisualAppearance().Add( 0x22,
COleVariant(_T("D:\\Temp\\ExRollist.Help\\select.ebn")) );
m_xmlgrid.SetSelBackColor( RGB(0,0,255) | 0x22000000 );
```

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

```
With AxRollist1
    With .VisualAppearance
        .Add(&H22, "D:\\Temp\\ExRollist.Help\\select.ebn")
    End With
    .Template = "SelBackColor = 587137024"
End With
```

where the 587137024 value is the hexa representation of 0x22FF0000

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

```
axRollist1.VisualAppearance.Add(0x22, "d:\\temp\\ExRollist.Help\\select.ebn");
axRollist1.Template = "SelBackColor = 587137024";
```

where the 587137024 value is the hexa representation of 0x22FF0000.

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

```
With thisform.Rollist1
    With .VisualAppearance
        .Add(34, "D:\\Temp\\ExRollist.Help\\select.ebn")
    EndWith
    .SelBackColor = RGB(0,0,255) + 570425344
EndWith
```


method Rollist.BeginUpdate ()

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

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

The BeginUpdate method prevents the control from painting until the [EndUpdate](#) method is called.

The following sample prevents control from painting while user adds a new item:

```
With Rollist1
    .BeginUpdate
    With .Add("New <b>item</b>", 1)
        .BackColor = vbBlue
        .ForeColor = vbWhite
    End With
    .EndUpdate
End With
```

property Rollist.BorderColor as Color

Specifies a value that indicates the color of the border for each item.

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

Use the BorderColor property to change the color for item's border. Use the [ItemBorderSize](#) property to define the size of the item's border.

property Rollist.BorderHeight as Long

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

Type	Description
Long	A long expression that defines the height of the empty borders at the top and bottom sides of the control, in pixels.

Use the BorderHeight and [BorderWidth](#) properties to define the size of the border around the control. By default, the BorderHeight property is 2 pixels.

property Rollist.BorderWidth as Long

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

Type	Description
Long	A long expression that indicates the width of the border at the left and right sides of the control, in pixels.

Use the [BorderHeight](#) and BorderWidth properties to define the size of the border around the control. By default, the BorderWidth property is 2 pixels.

property Rollist.Count as Long

Returns the number of items in the control.

Type	Description
Long	A long expression that indicates the number of items in the control.

The Count property counts the number of items in the control. Use the [Add](#) method to add new items to the control. Use the [Remove](#) method to remove items from the control.

property Rollist.Enabled as Boolean

Enables or disables the control.

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

Use the Enabled property to disable the control.

method Rollist.EndUpdate ()

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

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

property Rollist.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 Rollist.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:

```
Debug.Print Rollist1.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 Rollist.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 change the control's font.

property Rollist.ForeColor as Color

Specifies the control's foreground color.

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

The ForeColor property defines the default foreground color for items. Use the [ForeColor](#) property to define the foreground color for a givem item. Use the the <fgcolor> or <bgcolor> HTML tags in [Caption](#) property to define colored parts in the caption of the item.

property Rollist.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 Rollist.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"

<COLUMN1>.HTMLCaption = "A <img>pic1</img>"
<COLUMN2>.HTMLCaption = "B <img>pic2</img>"
<COLUMN3>.HTMLCaption = "A <img>pic1</img> + B <img>pic2</img>"
```


property Rollist.hWnd as Long

Retrieves the control's window handle.

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

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

method Rollist.Images (Handle as Variant)

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

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

The Handle parameter can be:

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

Handle as Variant

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

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

The following sample shows how to replace the entire list of icons, using a Microsoft Image List control (ImageList1):

```
Rollist1.Images ImageList1.hImageList
```

property Rollist.ImageSize as Long

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

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

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

property Rollist.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 item being accessed.
Item	An Item object being retrieved.

Use the Item property to access to the items of the control. Use the [Count](#) property to get the number of items in the control. Use the [VisibleItems](#) property to get the number of visible items.

The following sample enumerates the items in the control, using for Item property:

```
Dim i As Long
With Rollist1
  For i = 0 To .Count - 1
    With .Item(i)
      Debug.Print .Caption
    End With
  Next
End With
```

The following sample enumerates the items in the control, using for each statement:

```
Dim i As EXROLLISTLibCtl.Item
For Each i In Rollist1
  Debug.Print i.Caption
Next
```

property Rollist.ItemBackColor as Color

Specifies the default background color for all items.

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

The ItemBackColor property defines the default background color for items. The [ForeColor](#) property defines the default foreground color for items. Use the [BackColor](#) property to specify the background color for a given item. Use the [ForeColor](#) property to specify the foreground color for a given item.

property Rollist.ItemBorderSize as Long

Specifies the size of the border for items.

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

Use the ItemBorderSize property to define the size of the item's border. By default, the ItemBorderSize property is 3 pixels. The [SelectBorderSize](#) property specifies the size of the border of the selected item.

property Rollist.ItemFromPoint (X as OLE_XPOS_PIXELS, Y as OLE_YPOS_PIXELS) as Long

Retrieves the item from cursor.

Type	Description
X as OLE_XPOS_PIXELS	A long expression that indicates the x-coordinate in client coordinates.
Y as OLE_YPOS_PIXELS	A long expression that indicates the y-coordinate in client coordinates.
Long	A long expression that indicates the index of item over cursor.

Use the ItemFromPoint property to get the item from cursor. The ItemFromPoint property requires client coordinates so, X and Y coordinates must be translated to client coordinates. For instance, in VB you need to use functions like Screen.TwipsPerPixelX and Screen.TwipsPerPixelY to get the client coordinates.

property Rollist.ItemHeight as Long

Specifies the height of the item in pixels. If it is 0 the control computes the required height.

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

The ItemHeight property defines the height for each item. If the ItemHeight property is 0, the control automatically resizes the items to let [VisibleItems](#) fit the control's client area. Use the [ItemWidth](#) property to specify the width for items.

property Rollist.ItemWidth as Long

Specifies the width of the item in pixels. If it is 0 the control computes the required width.

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

The ItemWidth property specify the width of the items. By default, the width of the items is 128. If the ItemWidth property is 0 gets the maximum size and resizes all items. Use the [ItemHeight](#) property to specify the height for the items.

property Rollist.Picture as IPictureDisp

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

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

Use the Picture property to load a picture on the control's background. Use the [PictureDisplay](#) property to arrange the picture on the control's background.

property Rollist.PictureBox as PictureBoxEnum

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

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

Use the [Picture](#) property to load a picture into the control's background. Use the PictureBox property to arrange how the control's picture is displayed on its background.

method Rollist.Refresh ()

Refreshes the control.

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

If the [ItemWidth](#) property is 0, the Refresh method resizes all items. Else, the control is invalidated.

method Rollist.Remove (Index as Variant)

Removes a specific item from the collection.

Type	Description
Index as Variant	A long expression that indicates the index of item being removed.

Use the Remove method to remove a specific item. Use the [RemoveAllItems](#) method to remove all items in the control. Use the [Refresh](#) method to refresh the control.

method Rollist.RemoveAllItems ()

Removes all items in the collection.

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

The RemoveAllItems method clears the items collection. Use the [Remove](#) method to remove a given item.

method Rollist.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	A long expression that indicates the icon's handle
Index as Variant	A long expression that indicates the index where icon is inserted

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

Use the 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.

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

```
i = Rollist1.Replacelcon( LoadPicture("d:\icons\help.ico").Handle), in this case the i specifies the index where the icon was added
```

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

```
i = Rollist1.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:

```
Rollist1.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:

```
Rollist1.Replacelcon 0, -1
```


method Rollist.RollDown ()

Selects the next item by rotating the list.

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

Use the RollDown method to rotate down the list. Use the [Select](#) property to get the selected item.

method Rollist.RollUp ()

Selects the previously item by rotating the list.

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

The RollUp method rotates up the list. Use the [Select](#) property to get the selected item.

property Rollist.RoundCorner as Boolean

Specifies a value that indicates whether the items have round corners.

Type	Description
Boolean	A boolean expression that indicates whether the items have round corners.

By default, the RoundCorner property is True. Use the [ItemBorderSize](#) property to define the size of the item's border.

property Rollist.Select as Long

Selects a new item.

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

Use the Select property to get the index of the selected item. The selected item is always displayed at the center of the control. Use the [Item](#) property to access an item. The [Select](#) event is fired when user changes the selected item.

property Rollist.SelectBorderSize as Long

Specifies a value that indicates the size of the border for selected item.

Type	Description
Long	A long expression that indicates the size of border of the selected item, in pixels.

Use the SelectBorderSize property to define the size of the border of the selected item. Use the [ItemBorderSize](#) property to define the size of the border for rest of items. By default, the SelectBorderSize property is 5 pixels.

property Rollist.ShowImageList as Boolean

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

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

The property has effect only at design time.

method Rollist.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:

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

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

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

<fgcolor=FFFFFF>outlined</fgcolor></out>" generates the following picture:



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



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



property Rollist.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 execute a template script and gets the result.

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

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

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

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

An instruction can be one of the following:

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

separated by commas. (Sample: `h = InsertItem(0,"New Child")`)

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

The Template supports the following general functions:

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

property Rollist.TemplateDef as Variant

Defines inside variables for the next Template/ExecuteTemplate call.

Type	Description
Variant	A string expression that indicates the Dim declaration, or any Object expression to be assigned to previously declared variables.

The TemplateDef property has been added to allow programming languages such as dBASE Plus to set control's properties with multiple parameters. It is known that programming languages such as **dBASE Plus** or **XBasic from AlphaFive**, does not support setting a property with multiple parameters. In other words, these programming languages does not support something like *Property(Parameters) = Value*, so our controls provide an alternative using the TemplateDef method. The first call of the TemplateDef should be a declaration such as "Dim a,b" which means the next 2 calls of the TemplateDef defines the variables a and b. The next call should be [Template](#) or [ExecuteTemplate](#) property which can use the variable a and b being defined previously.

So, calling the TemplateDef property should be as follows:

```
with (Control)
    TemplateDef = [Dim var_Column]
    TemplateDef = var_Column
    Template = [var_Column.Def(4) = 255]
endwith
```

This sample allocates a variable var_Column, assigns the value to the variable (the second call of the TemplateDef), and the Template call uses the var_Column variable (as an object), to call its Def property with the parameter 4.

Let's say we need to define the background color for a specified column, so we need to call the Def(exCellBackColor) property of the column, to define the color for all cells in the column.

The following **VB6** sample shows setting the Def property such as:

```
With Control
    .Columns.Add("Column 1").Def(exCellBackColor) = 255
    .Columns.Add "Column 2"
    .Items.AddItem 0
    .Items.AddItem 1
```

```
.Items.AddItem 2  
End With
```

In **dBASE Plus**, calling the Def(4) has no effect, instead using the TemplateDef helps you to use properly the Def property as follows:

```
local Control,var_Column  
  
Control = form.ActiveX1.nativeObject  
// Control.Columns.Add("Column 1").Def(4) = 255  
var_Column = Control.Columns.Add("Column 1")  
with (Control)  
    TemplateDef = [Dim var_Column]  
    TemplateDef = var_Column  
    Template = [var_Column.Def(4) = 255]  
endwith  
Control.Columns.Add("Column 2")  
Control.Items.AddItem(0)  
Control.Items.AddItem(1)  
Control.Items.AddItem(2)
```

The equivalent sample for **XBasic in A5**, is as follows:

```
Dim Control as P  
Dim var_Column as P  
  
Control = topparent:CONTROL_ACTIVEX1.activex  
' Control.Columns.Add("Column 1").Def(4) = 255  
var_Column = Control.Columns.Add("Column 1")  
Control.TemplateDef = "Dim var_Column"  
Control.TemplateDef = var_Column  
Control.Template = "var_Column.Def(4) = 255"  
  
Control.Columns.Add("Column 2")  
Control.Items.AddItem(0)  
Control.Items.AddItem(1)  
Control.Items.AddItem(2)
```

The samples just call the `Column.Def(4) = Value`, using the `TemplateDef`. The first call of `TemplateDef` property is `"Dim var_Column"`, which indicates that the next call of the `TemplateDef` will defines the value of the variable `var_Column`, in other words, it defines the object `var_Column`. The last call of the `Template` property uses the `var_Column` member to use the x-script and so to set the `Def` property so a new color is being assigned to the column.

The `TemplateDef`, [Template](#) and [ExecuteTemplate](#) support x-script language (`Template` script of the `Exontrols`), like explained bellow:

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

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

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

The x-script may uses constant expressions as follow:

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

make sure that it is different than ' which allows adding comments inline. *Sample: "text" indicates the string text.*

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

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

method Rollist.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 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 Rollist.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 [ShowToolTip](#) method to display a custom tooltip.

property Rollist.ToolTipFont as IFontDisp

Retrieves or sets the tooltip's font.

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

Use the ToolTipFont property to assign a font for the control's tooltip. The [ToolTipPopDelay](#) property specifies the period in ms of time the ToolTip remains visible if the mouse pointer is stationary within a control. Use the [ToolTipWidth](#) property to specify the width of the tooltip window.

property Rollist.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 [ShowToolTip](#) method to display a custom tooltip.

property Rollist.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 [ShowToolTip](#) method to display a custom tooltip.



property Rollist.UseHandCursor as Boolean

Specifies a value that indicates whether the control displays the hand cursor when cursor is over the items.

Type	Description
Boolean	A boolean expression that specifies the control displays the hand cursor when cursor is over the items.

The UseHandCursor property specifies whether the control displays the hand cursor when the the cursor is over an item. By default, the UseHandCursor property is True.

property Rollist.Version as String

Retrieves the control's version.

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

The Version property specifies the control's version.

property Rollist.VisibleItems as Long

Specifies a value that indicates the count of visible items.

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

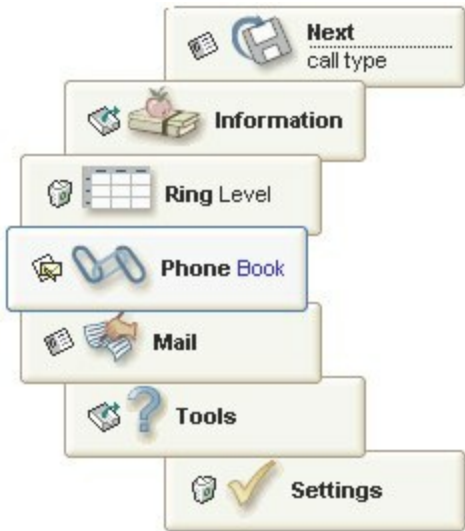
Use the VisibleItems property to specify the number of visible items. By default, the VisibleItems property is 7. if the number is not odd and less than 3, an error occurs when changing the VisibleItems property. Use the [Add](#) method to add new items to the control.

property Rollist.VisualAppearance as Appearance

Retrieves the control's appearance.

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

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



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

- selected item, [Background](#) property
- item, [BackColor](#) property
- items, [ItemBackColor](#) propetry

ExRollist 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: {08EA793D-92E4-4DA3-A66D-06E33F1F3945}. The object's program identifier is: "Exontrol.Rollist". The /COM object module is: "ExRollist.dll"

The ExRollist component supports the following events:

Name	Description
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.
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.
Roll	Occurs when the control rotates the items.
Select	Occurs when the user selects an item.

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,
AxEXROLLISTLib._IRollistEvents_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 :
WidesString);
begin
end;
```

**Delphi 8
(.NET
only)**

```
procedure AnchorClick(sender: System.Object; e:
AxEXROLLISTLib._IRollistEvents_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
AxEXROLLISTLib._IRollistEvents_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(oRollist,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.

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

```
C# private void Click(object sender)
{
}
```

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

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

```
C# private void ClickEvent(object sender, EventArgs e)
{
}
```

```
C++ void OnClick()
{
}
```

```
C++ Builder void __fastcall Click(TObject *Sender)
{
}
```

```
Delphi procedure Click(ASender: TObject; );
begin
end;
```

```
Delphi 8 (.NET only) procedure ClickEvent(sender: System.Object; e: System.EventArgs);
begin
end;
```


Powe... begin event Click()
end event Click

VB.NET Private Sub ClickEvent(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles ClickEvent
End Sub

VB6 Private Sub Click()
End Sub

VBA Private Sub Click()
End Sub

VFP LPARAMETERS nop

Xbas... PROCEDURE OnClick(oRollist)
RETURN

Syntax for Click event, **ICOM** 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 a cell has been double-clicked.

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, AxEXROLLISTLib._IRollistEvents_DbtClickEvent
e)
{
}
```

C++

```
void OnDbtClick(short Shift,long X,long Y)
{
}
```

C++ Builder

```
void __fastcall DbtClick(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:  
AxEXROLLISTLib._IRollistEvents_DblClickEvent);  
begin  
end;
```

Powe...

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

VB.NET

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

VB6

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

VBA

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

VFP

```
LPARAMETERS Shift,X,Y
```

Xbas...

```
PROCEDURE OnDblClick(oRollist,Shift,X,Y)  
RETURN
```

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

Java...

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

VBSc...

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

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

Visual
Data...

```
Procedure OnComDbClick Short IIShift OLE_XPOS_PIXELS IIX OLE_YPOS_PIXELS  
IYY  
    Forward Send OnComDbClick IIShift IIX IYY  
End_Procedure
```

Visual
Objects

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

X++

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

XBasic

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

dBASE

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

The following sample finds the item that was dbl clicked:

```
Private Sub Rollist1_DbClick(Shift As Integer, X As Single, Y As Single)  
    Dim i As Long  
    With Rollist1  
        i = .ItemFromPoint(X / Screen.TwipsPerPixelX, Y / Screen.TwipsPerPixelY)  
        If (i >= 0) Then  
            With .Item(i)  
                Debug.Print .Caption  
            End With  
        End If  
    End With  
End Sub
```

event KeyDown (KeyCode as Integer, Shift as Integer)

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

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

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

```
ShiftDown = (Shift And 1) > 0
CtrlDown = (Shift And 2) > 0
AltDown = (Shift And 4) > 0
```

In a procedure, you can test for any combination of conditions, as in this example:
If AltDown And CtrlDown Then

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

C#

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

VB

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

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

C#

```
private void KeyDownEvent(object sender,
AxEXROLLISTLib._IRollistEvents_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:  
AxEXROLLISTLib._IRollistEvents_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  
AxEXROLLISTLib._IRollistEvents_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(oRollist,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

The following sample starts prints the key code:

```
Private Sub Rollist1_KeyDown(KeyCode As Integer, Shift As Integer)
    Debug.Print "KeyDown = " & KeyCode
End Sub
```


event KeyPress (KeyAscii as Integer)

Occurs when the user presses and releases an ANSI key.

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

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

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

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

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

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

```
C# private void KeyPressEvent(object sender,
AxEXROLLISTLib._IRollistEvents_KeyPressEvent e)
{
}
```

```
C++ void OnKeyPress(short FAR* KeyAscii)
{
}
```

```
C++ Builder void __fastcall KeyPress(TObject *Sender,short * KeyAscii)
{
}
```

Delphi procedure KeyPress(ASender: TObject; var KeyAscii : Smallint);
begin
end;

**Delphi 8
(.NET
only)** procedure KeyPressEvent(sender: System.Object; e:
AxEXROLLISTLib._IRollistEvents_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
AxEXROLLISTLib._IRollistEvents_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(oRollist,KeyAscii)
RETURN

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

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

VBSc... <SCRIPT LANGUAGE="VBScript">
Function KeyPress(KeyAscii)
End Function
</SCRIPT>

Visual
Data...

```
Procedure OnComKeyPress Short Integer KeyAscii  
    Forward Send OnComKeyPress Integer KeyAscii  
End_Procedure
```

Visual
Objects

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

C++

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

XBasic

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

dBASE

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

event KeyUp (KeyCode as Integer, Shift as Integer)

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

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

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

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

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

VBPrivate Sub KeyUp(ByVal sender As System.Object,ByRef KeyCode As Short,ByVal Shift As Short) Handles KeyUpEnd Sub

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

C#private void KeyUpEvent(object sender,AxEXROLLISTLib._IRollistEvents_KeyUpEvent e){}

C++void OnKeyUp(short FAR* KeyCode,short Shift){}

C++ Buildervoid __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:  
AxEXROLLISTLib._IRollistEvents_KeyUpEvent);  
begin  
end;
```

Powe...

```
begin event KeyUp(integer KeyCode,integer Shift)  
end event KeyUp
```

VB.NET

```
Private Sub KeyUpEvent(ByVal sender As System.Object, ByVal e As  
AxEXROLLISTLib._IRollistEvents_KeyUpEvent) Handles KeyUpEvent  
End Sub
```

VB6

```
Private Sub KeyUp(KeyCode As Integer,Shift As Integer)  
End Sub
```

VBA

```
Private Sub KeyUp(KeyCode As Integer,ByVal Shift As Integer)  
End Sub
```

VFP

```
LPARAMETERS KeyCode,Shift
```

Xbas...

```
PROCEDURE OnKeyUp(oRollist,KeyCode,Shift)  
RETURN
```

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

Java...

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

VBSc...

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

```
</SCRIPT>
```

Visual
Data...

```
Procedure OnComKeyUp Short 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.

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,
AxEXROLLISTLib._IRollistEvents_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: AxEXROLLISTLib._IRollistEvents_MouseDownEvent);
begin
end;

PowerBuilder 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 AxEXROLLISTLib._IRollistEvents_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

Xbase++ PROCEDURE OnMouseDown(oRollist,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.Rollist.1::OLE_XPOS_PIXELS,Y as
OLE::Exontrol.Rollist.1::OLE_YPOS_PIXELS)
end function

dBASE function nativeObject_MouseDown(Button,Shift,X,Y)
return

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.

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 MouseEventArgs
End Sub
```

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

```
C# private void MouseEventArgsEvent(object sender,
AxEXROLLISTLib._IRollistEvents_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:
AxEXROLLISTLib._IRollistEvents_MouseMoveEvent);
begin
end;
```

Powe...

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

VB.NET

```
Private Sub MouseMoveEvent(ByVal sender As System.Object, ByVal e As
AxEXROLLISTLib._IRollistEvents_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(oRollist,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.Rollist.1::OLE_XPOS_PIXELS,Y as OLE::Exontrol.Rollist.1::OLE_YPOS_PIXELS) end function</pre>
dBASE	<pre>function nativeObject_MouseMove(Button,Shift,X,Y) return</pre>

The following sample prints the item over the cursor:

```
Private Sub Rollist1_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As
Single)
    Dim i As Long
    With Rollist1
        i = .ItemFromPoint(X / Screen.TwipsPerPixelX, Y / Screen.TwipsPerPixelY)
```

```
If (i >= 0) Then
    With .Item(i)
        Debug.Print .Caption
    End With
End If
End With
End Sub
```

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.

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

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

VBPrivate 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,
AxEXROLLISTLib._IRollistEvents_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:
AxEXROLLISTLib._IRollistEvents_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
AxEXROLLISTLib._IRollistEvents_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(oRollist,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.Rollist.1::OLE_XPOS_PIXELS,Y as
OLE::Exontrol.Rollist.1::OLE_YPOS_PIXELS)
end function`

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

event Roll (OldItem as Long, NewItem as Long)

Occurs when the control rotates the items.

Type	Description
OldItem as Long	A long expression that indicates the index of item being deactivated.
NewItem as Long	A long expression that indicates the index of item being activated.

Use the Roll event to notify your application that the control rotates the list and a new item is displayed on the center of the control. Use the [Select](#) event to notify your application that user selects an item by keyboard or by mouse.

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

C#

```
private void Roll(object sender,int OldItem,int NewItem)
{
}
```

VB

```
Private Sub Roll(ByVal sender As System.Object,ByVal OldItem As Integer,ByVal
NewItem As Integer) Handles Roll
End Sub
```

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

C#

```
private void Roll(object sender, AxEXROLLISTLib._IRollistEvents_RollEvent e)
{
}
```

C++

```
void OnRoll(long OldItem,long NewItem)
{
}
```

C++ Builder

```
void __fastcall Roll(TObject *Sender,long OldItem,long NewItem)
{
}
```

Delphi

```
procedure Roll(ASender: TObject; OldItem : Integer;NewItem : Integer);
begin
```

```
end;
```

Delphi 8
(.NET
only)

```
procedure Roll(sender: System.Object; e:  
AxEXROLLISTLib._IRollistEvents_RollEvent);  
begin  
end;
```

Powe...

```
begin event Roll(long OldItem,long NewItem)  
end event Roll
```

VB.NET

```
Private Sub Roll(ByVal sender As System.Object, ByVal e As  
AxEXROLLISTLib._IRollistEvents_RollEvent) Handles Roll  
End Sub
```

VB6

```
Private Sub Roll(ByVal OldItem As Long,ByVal NewItem As Long)  
End Sub
```

VBA

```
Private Sub Roll(ByVal OldItem As Long,ByVal NewItem As Long)  
End Sub
```

VFP

```
LPARAMETERS OldItem,NewItem
```

Xbas...

```
PROCEDURE OnRoll(oRollist,OldItem,NewItem)  
RETURN
```

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

Java...

```
<SCRIPT EVENT="Roll(OldItem,NewItem)" LANGUAGE="JScript">  
</SCRIPT>
```

VBSc...

```
<SCRIPT LANGUAGE="VBScript">  
Function Roll(OldItem,NewItem)  
End Function  
</SCRIPT>
```

Visual
Data...

```
Procedure OnComRoll Integer IIOldItem Integer IINewItem  
Forward Send OnComRoll IIOldItem IINewItem
```

End_Procedure

Visual
Objects

METHOD OCX_Roll(OldItem,NewItem) CLASS MainDialog
RETURN NIL

X++

```
void onEvent_Roll(int _OldItem,int _NewItem)
{
}
```

XBasic

```
function Roll as v (OldItem as N,NewItem as N)
end function
```

dBASE

```
function nativeObject_Roll(OldItem,NewItem)
return
```

The following sample highlights the item in the center of the control (selected item):

```
Private Sub Rollist1_Roll(ByVal OldItem As Long, ByVal NewItem As Long)
    With Rollist1(OldItem)
        .ClearBackColor
        .ClearForeColor
    End With
    With Rollist1(NewItem)
        .BackColor = vbBlue
        .ForeColor = vbWhite
    End With
End Sub
```

event Select ()

Occurs when the user selects an item.

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

Use the Select event to notify your application that the user selects an item by keyboard or by mouse. Use the [Select](#) property to retrieve the index of selected item.

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

C#

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

VB

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

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

C#

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

C++

```
void OnSelect()
{
}
```

C++ Builder

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

Delphi

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

Delphi 8 (.NET only)

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

Powe...

```
begin event Select()  
end event Select
```

VB.NET

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

VB6

```
Private Sub Select()  
End Sub
```

VBA

```
Private Sub Select()  
End Sub
```

VFP

```
LPARAMETERS nop
```

Xbas...

```
PROCEDURE OnSelect(oRollist)  
RETURN
```

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

Java...

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

VBSc...

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

Visual
Data...

```
Procedure OnComSelect  
Forward Send OnComSelect  
End_Procedure
```

Visual
Objects

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

X++

```
void onEvent_Select()  
{  
}
```

XBasic

```
function Select as v ()  
end function
```

dBASE

```
function nativeObject_Select()  
return
```

The following sample displays the caption of item being selected:

```
Private Sub Rollist1_Select()  
    With Rollist1  
        With .Item(.Select)  
            Debug.Print .Caption  
        End With  
    End With  
End Sub
```