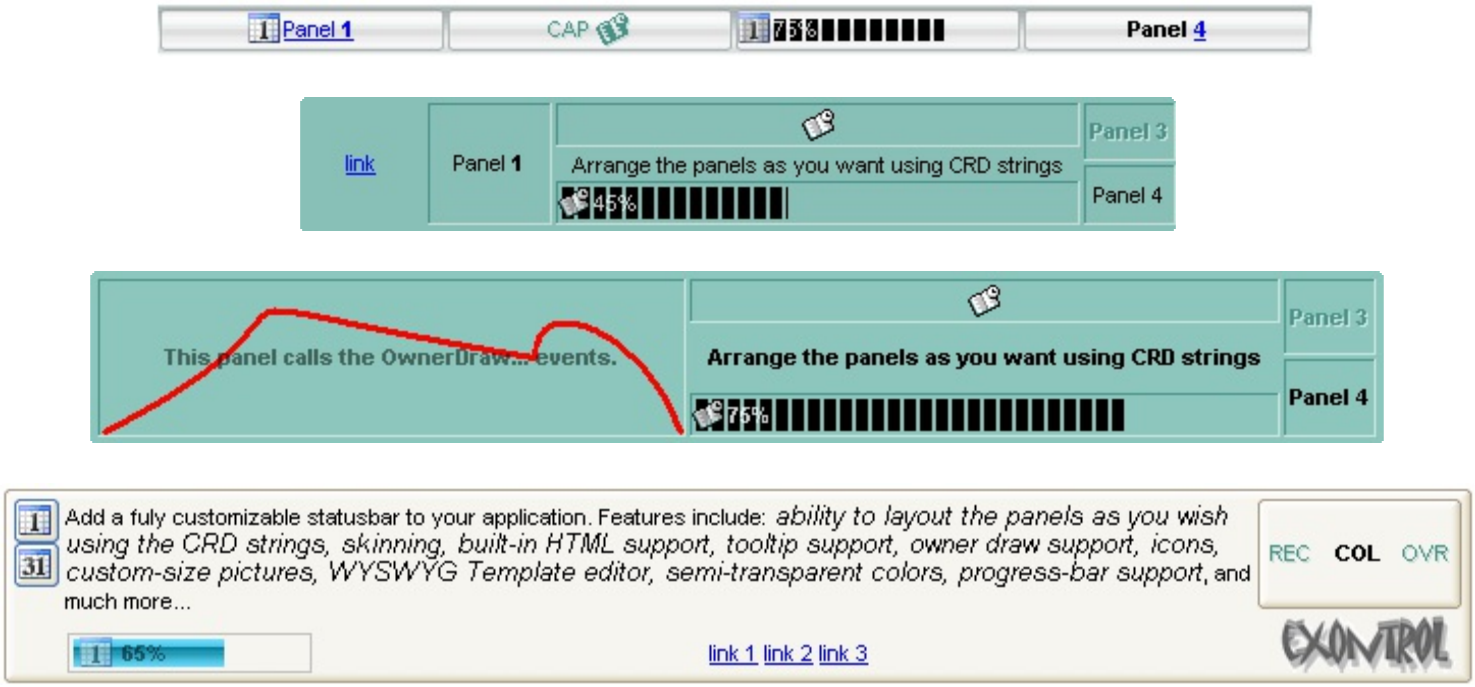
ExStatusBar

The Exontrol's ExStatusBar component provides statusbar panels to your forms. The statusbar is a component (widget) often found at the bottom of windows in a graphical user interface. It is very frequently divided into sections, each of which shows different information. Its job is primarily to display information about the current state of its window, although some status bars have extra functionality. Usually, the status bar often called a status line in this context displays the current state of the application, and helpful keyboard shortcuts.

Features include:

- **Ability to layout the panels as you wish using the CRD strings.**
- **Unlimited** options to show any HTML text, images, colors, EBNs, patterns, frames anywhere on the panel's background
- Skinnable Interface support (ability to apply a skin to the any background part, including the scroll bars)
- **ActiveX hosting** (ability to insert any ActiveX controls to any panel)
- WYSWYG Template/Layout Editor support
- Built-in HTML support
- Semi-transparent colors support
- Progress-bar support
- Owner draw support
- Multiple lines HTML Tooltip support
- icons, custom-size pictures



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

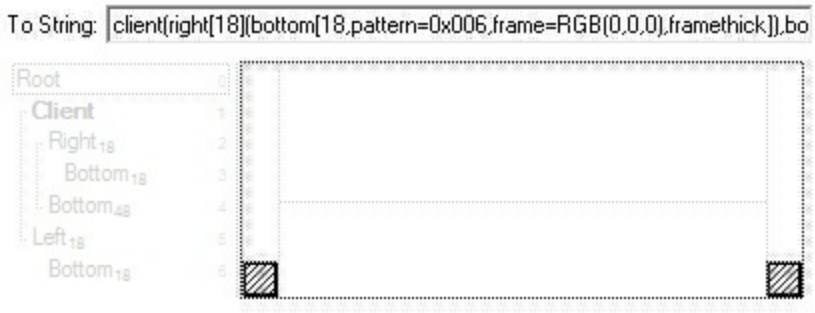
constants AppearanceEnum

The AppearanceEnum enumeration specifies the appearance of the control's border. Use the Appearance property to specify the control's border. The AppearanceEnum type supports the following values:

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

constants BackgroundExtPropertyEnum

The BackgroundExtPropertyEnum type specifies the UI properties of the part of the EBN you can access/change at runtime. The BackgroundExt property specifies the EBN String format to be displayed on the event's background. The BackgroundExtValue property access the value of the giving property for specified part of the EBN. The BackgroundExtPropertyEnum type supports the following values:

Name	Value	Description
exToStringExt	0	Specifies the part's ToString representation. The BackgroundExt property specifies the EBN String format to be displayed on the object's background. *The Exontrol's eXButton WYSWYG Builder helps you to generate or view the EBN String Format, in the **To String** field.* *Sample:* *"client(right[18](bottom[18,pattern=6,frame=0,framethick]),bottom[4(bottom[18,pattern=6,frame=0,framethick])"* *generates the following layout:*  *where it is applied to an object it looks as follows:*  *(String expression, read-only).*

exBackColorExt	1	Indicates the background color / EBN color to be shown on the part of the object. *Sample: 255 indicates red, RGB(0,255,0) green, or 0x1000000.* *(Color/Numeric expression, The last 7 bits in the high significant byte of the color indicate the identifier of the skin being used)*
		Specifies the position/size of the object, depending on the object's anchor. The syntax of the exClientExt is related to the exAnchorExt value. *For instance, if the object is anchored to the left side of the parent (exAnchorExt = 1), the exClientExt specifies just the width of the part in pixels/percents, not including the position. In case, the exAnchorExt is client, the exClientExt has no effect.* *Based on the exAnchorExt value the exClientExt is:* • *0 (**none**, the object is not anchored to any side), the format of the exClientExt is **"left,top,width,height"** (as string) where (left,top) margin indicates the position where the part starts, and the (width,height) pair specifies its size. The left, top, width or height could be any expression (+,-,/ or *) that can include numbers associated with pixels or percents. For instance: "25%,25%,50%,50%" indicates the middle of the parent object, and so when the parent is resized the client is resized accordingly. The "50%-8,50%-8,16,16" value specifies that the size of the object is always 16x16 pixels and positioned on the center of the parent object.* • *1 (**left**, the object is anchored to left side of the parent), the format of the exClientExt is **width** (string or numeric) where width indicates the width of the object in pixels, percents or a combination of them using +,-,/ or * operators. For instance: "50%" indicates*

exClientExt	2	*the half of the parent object, and so when the parent is resized the client is resized accordingly. The 16 value specifies that the size of the object is always 16 pixels.* • *2 (**right**, the object is anchored to right side of the parent object), the format of the exClientExt is **width** (string or numeric) where width indicates the width of the object in pixels, percents or a combination of them using +,-,/ or * operators. For instance: "50%" indicates the half of the parent object, and so when the parent is resized the client is resized accordingly. The 16 value specifies that the size of the object is always 16 pixels.* • *3 (**client**, the object takes the full available area of the parent), the exClientExt has no effect.* • *4 (**top**, the object is anchored to the top side of the parent object), the format of the exClientExt is **height** (string or numeric) where height indicates the height of the object in pixels, percents or a combination of them using +,-,/ or * operators. For instance: "50%" indicates the half of the parent object, and so when the parent is resized the client is resized accordingly. The 16 value specifies that the size of the object is always 16 pixels.* • *5 (**bottom**, the object is anchored to bottom side of the parent object), the format of the exClientExt is **height** (string or numeric) where height indicates the height of the object in pixels, percents or a combination of them using +,-,/ or * operators. For instance: "50%" indicates the half of the parent object, and so when the parent is resized the client is resized accordingly. The 16 value specifies that the size of the object is always 16 pixels.* *Sample: 50% indicates half of the parent, 25 indicates 25 pixels, or 50%-8 indicates 8-pixels left from the center of the parent.*

		(String/Numeric expression)
exAnchorExt	3	Specifies the object's alignment relative to its parent. *The valid values for exAnchorExt are:* • *0 **(none)**, the object is not anchored to any side,* • *1 **(left)**, the object is anchored to left side of the parent,* • *2 **(right)**, the object is anchored to right side of the parent object,* • *3 **(client)**, the object takes the full available area of the parent,* • *4 **(top)**, the object is anchored to the top side of the parent object,* • *5 **(bottom)**, the object is anchored to bottom side of the parent object* *(Numeric expression)*
		Specifies the HTML text to be displayed on the object. *The exTextExt supports the following built-in HTML tags:* • ** ... ** displays the text in **bold** • ** ... ** displays the text in *italics* • **<u> ... </u>** underlines the text • **<s> ... </s>** ~~Strike~~-through text • **<a id;options> ... ** displays an anchor element that can be clicked. An anchor is a piece of text or some other object (for example an image) which marks the beginning and/or the end of a hypertext link.The <a> element is used to mark that piece of text (or inline image), and to give its hypertextual relationship to other documents. The control fires the *AnchorClick(AnchorID, Options)* event when the user clicks the anchor element. The

FormatAnchor property customizes the visual effect for anchor elements.

- ** ... ** displays portions of text with a different font and/or different size. For instance, the "bit" draws the bit text using the Tahoma font, on size 12 pt. If the name of the font is missing, and instead size is present, the current font is used with a different size. For instance, "bit" displays the bit text using the current font, but with a different size.
- **<fgcolor rrggbb> ... </fgcolor>** or <fgcolor=rrggbb> ... </fgcolor> displays text with a specified foreground color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<bgcolor rrggbb> ... </bgcolor>** or <bgcolor=rrggbb> ... </bgcolor> displays text with a specified background color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<solidline rrggbb> ... </solidline>** or <solidline=rrggbb> ... </solidline> draws a solid-line on the bottom side of the current text-line, of specified RGB color. The <solidline> ... </solidline> draws a black solid-line on the bottom side of the current text-line. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<dotline rrggbb> ... </dotline>** or <dotline=rrggbb> ... </dotline> draws a dot-line on the bottom side of the current text-line, of specified RGB color. The <dotline> ... </dotline> draws a black dot-line on the bottom side of the current text-line. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<upline> ... </upline>** draws the line on the top side of the current text-line (requires <solidline> or <dotline>).
- **<r>** right aligns the text

exTextExt	4	• **<c>** centers the text • ** ** forces a line-break • **<img>number[:width]</img>** inserts an icon inside the text. The number indicates the index of the icon being inserted. Use the Images method to assign a list of icons to your chart. The last 7 bits in the high significant byte of the number expression indicates the identifier of the skin being used to paint the object. Use the Add method to add new skins to the control. If you need to remove the skin appearance from a part of the control you need to reset the last 7 bits in the high significant byte of the color being applied to the part. The width is optional and indicates the width of the icon being inserted. Using the width option you can overwrite multiple icons getting a nice effect. By default, if the width field is missing, the width is 18 pixels. • **<img>key[:width]</img>** inserts a custom size picture into the text being previously loaded using the HTMLPicture property. The Key parameter indicates the key of the picture being displayed. The Width parameter indicates a custom size, if you require to stretch the picture, else the original size of the picture is used. • **&** glyph characters as **&** (&), **<** (<), **>** (>), **&qout;** (") and **&#number;** (the character with specified code), For instance, the € displays the EUR character. The **&** ampersand is only recognized as markup when it is followed by a known letter or a #character and a digit. For instance if you want to display bold in HTML caption you can use bold • **<off offset> ... </off>** defines the vertical offset to display the text/element. The offset parameter defines the offset to display the element. This tag is inheritable, so the offset is keep while the associated </off> tag is found. You can use the <off offset> HTML tag in combination with the to define

a smaller or a larger font to be displayed. For instance: "Text with <**off** 6>subscript" displays the text such as: Text with $_{\text{subscript}}$ The "Text with <**off** -6>superscript" displays the text such as: Text with ^{subscript}

- **<gra rrggbb;mode;blend> ... </gra>** defines a gradient text. The text color or <fgcolor> defines the starting gradient color, while the rr/gg/bb represents the red/green/blue values of the ending color, 808080 if missing as gray. The mode is a value between 0 and 4, 1 if missing, and blend could be 0 or 1, 0 if missing. The HTML tag can be used to define the height of the font. Any of the rrggbb, mode or blend field may not be specified. The <gra> with no fields, shows a vertical gradient color from the current text color to gray (808080). For instance the "<**gra** FFFFFF;1;1>gradient-center</**gra**>" generates the following picture:

  gradient-center

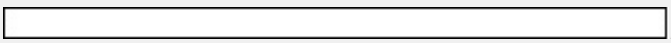
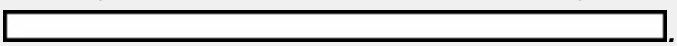
- **<out rrggbb;width> ... </out>** shows the text with outlined characters, where rr/gg/bb represents the red/green/blue values of the outline color, 808080 if missing as gray, width indicates the size of the outline, 1 if missing. The text color or <fgcolor> defines the color to show the inside text.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

  outlined

- **<sha rrggbb;width;offset> ... </sha>** define a text with a shadow, where rr/gg/bb represents the red/green/blue values of the shadow color, 808080 if missing as gray, width indicates the size of shadow, 4 if missing, and offset indicates the offset from the origin to display the text's shadow, 2 if missing. The text

		color or <fgcolor> defines the color to show the inside text. The HTML tag can be used to define the height of the font. For instance the "<**sha**>shadow</**sha**> " generates the following picture: shadow or "*<**sha** 404040;5;0> <fgcolor=FFFFFF>outline anti-aliasing</fgcolor></**sha**>*" gets: outline anti-aliasing *(String expression)*
exTextExtWordWrap	5	Specifies that the object is wrapping the text. The exTextExt value specifies the HTML text to be displayed on the part of the EBN object. This property has effect only if there is a text assigned to the part using the exTextExt flag. *(Boolean expression)*
		Indicates the alignment of the text on the object. The exTextExt value specifies the HTML text to be displayed on the part of the EBN object. This property has effect only if there is a text assigned to the part using the exTextExt flag. *The valid values for exTextExtAlignment are:* • *0, (hexa 0x00, **Top-Left**), Text is vertically aligned at the top, and horizontally aligned on the left.* • *1, (hexa 0x01, **Top-Center**), Text is vertically aligned at the top, and horizontally aligned at the center.* • *2, (hexa 0x02, **Top-Right**), Text is vertically aligned at the top, and horizontally aligned on the right.* • *16, (hexa 0x10, **Middle-Left**), Text is*

exTextExtAlignment	6	*vertically aligned in the middle, and horizontally aligned on the left.* • *17, (hexa 0x11, **Middle-Center**), Text is vertically aligned in the middle, and horizontally aligned at the center.* • *18, (hexa 0x12, **Middle-Right**), Text is vertically aligned in the middle, and horizontally aligned on the right.* • *32, (hexa 0x20, **Bottom-Left**), Text is vertically aligned at the bottom, and horizontally aligned on the left.* • *33, (hexa 0x21, **Bottom-Center**), Text is vertically aligned at the bottom, and horizontally aligned at the center.* • *34, (hexa 0x22, **Bottom-Right**), Text is vertically aligned at the bottom, and horizontally aligned on the right.* *(Numeric expression)*
		Indicates the pattern to be shown on the object. The exPatternColorExt specifies the color to show the pattern. *The valid values for exPatternExt are:* • *0, (hexa 0x000, **Empty**), The pattern is not visible* • *1, (hexa 0x001, **Solid**),* • *2, (hexa 0x002, **Dot**),* • *3, (hexa 0x003, **Shadow**),* • *4, (hexa 0x004, **NDot**),* • *5, (hexa 0x005, **FDiagonal**),* • *6, (hexa 0x006, **BDiagonal**),* • *7, (hexa 0x007, **DiagCross**),*

exPatternExt	7	• *8, (hexa 0x008, **Vertical**),*  • *9, (hexa 0x009, **Horizontal**),*  • *10, (hexa 0x00A, **Cross**),*  • *11, (hexa 0x00B, **Brick**),*  • *12, (hexa 0x00C, **Yard**),*  • *256, (hexa 0x100, **Frame**),*  *. The exFrameColorExt specifies the color to show the frame. The Frame flag can be combined with any other flags.* • *768, (hexa 0x300, **FrameThick**),*  *. The exFrameColorExt specifies the color to show the frame. The Frame flag can be combined with any other flags.* *(Numeric expression)*
exPatternColorExt	8	Indicates the color to show the pattern on the object. The exPatternColorExt property has effect only if the exPatternExt property is not 0 (empty). The exFrameColorExt specifies the color to show the frame (the exPatternExt property includes the exFrame or exFrameThick flag) *(Color expression)*
exFrameColorExt	9	Indicates the color to show the border-frame on the object. This property set the Frame flag for exPatternExt property. *(Color expression)*
exFrameThickExt	10	Specifies that a thick-frame is shown around the object. This property set the FrameThick flag for exPatternExt property.

		(Boolean expression)
exUserDataExt	11	Specifies an extra-data associated with the object. (*Variant expression)*

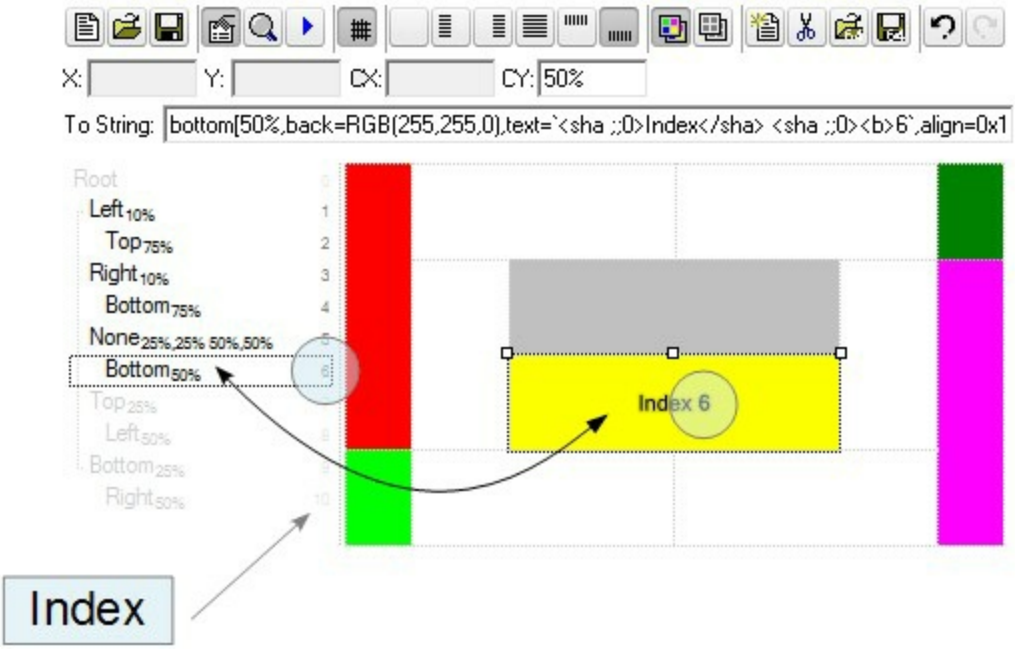
constants BackgroundPartEnum

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
exToolTipAppearance	64	Specifies the visual appearance of the borders of the tooltips.
exToolTipBackColor	65	Specifies the tooltip's background color.
exToolTipForeColor	66	Specifies the tooltip's foreground color.

constants IndexExtEnum

The IndexExtEnum type specifies the index of the part of the EBN object to be accessed. The Index parameter of the BackgroundExtValue property indicates the index of the part of the EBN object to be changed or accessed. *The Exontrol's eXButton WYSWYG Builder helps you to generate or view the EBN String Format, in the **To String** field. The list of objects that compose the EBN are displayed on the left side of the Builder tool, and the Index of the part is displayed on each item aligned to the right as shown in the following screen shot:*

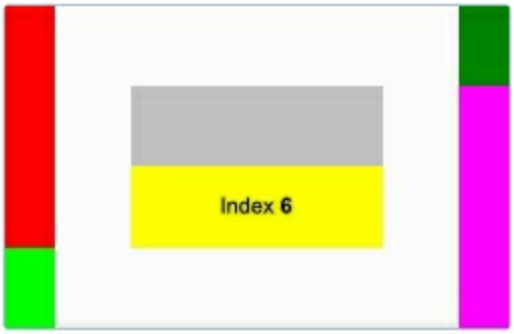


In this sample, there are 11 objects that composes the EBN, so the Index property goes from 0 which indicates the root, and 10, which is the last item in the list

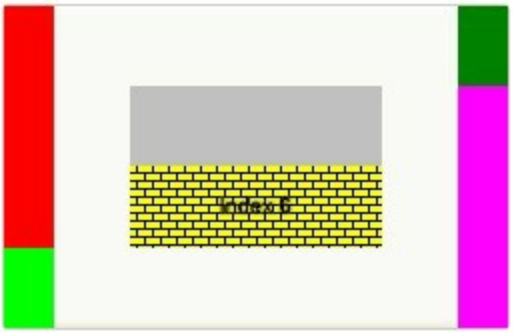
So, let's apply this format to an object, to change the exPatternExt property for the object with the Index 6:

Before calling the BodyBackgroundExt property:

After calling the BodyBackgroundExt property:



and now, let's change the exPatternExt property of the object with the Index 6 to 11 (Yard), so finally we got:



The IndexExtEnum type supports the following values:

Name	Value	Description
exIndexExtRoot	0	Specifies the part of the object with the index 0 (root).
exIndexExt1	1	Specifies the part of the object with the index 1.
exIndexExt2	2	Specifies the part of the object with the index 2.
exIndexExt3	3	Specifies the part of the object with the index 3.
exIndexExt4	4	Specifies the part of the object with the index 4.
exIndexExt5	5	Specifies the part of the object with the index 5.
exIndexExt6	6	Specifies the part of the object with the index 6.
exIndexExt7	7	Specifies the part of the object with the index 7.

constants PictureDisplayEnum

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

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

constants TextAlignEnum

The TextAlignEnum type specifies the possible alignments of the text inside the panel. Use the Alignment property to align the caption inside the panel.

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

Appearance object

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

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

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

Adds or replaces a skin object to the control.

Type	Description
ID as Long	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	A string expression that indicates: • an Windows XP Theme part, it should start with "XP:". For instance the "**XP:Header 1 2**" indicates the part 1 of the Header class in the state 2, in the current Windows XP theme. In this case the format of the Skin parameter should be: "XP: Control/ClassName Part State" where the ClassName defines the window/control class name in the Windows XP Theme, the Part indicates a long expression that defines the part, and the State indicates the state like listed at the end of the document. This option is available only on Windows XP that supports Themes API. • copy of another skin with different coordinates, if it begins with "CP:" . For instance, you may need to display a specified skin on a smaller rectangle. In this case, the string starts with "CP:", and contains the following "CP:n l t r b", where the n is the identifier being copied, the l, t, r, and b indicate the left, top, right and bottom coordinates being used to adjust the rectangle where the skin is displayed. For instance, the "**CP:1 4 0 -4 0**", indicates that the skin is displayed on a smaller rectangle like follows. Let's say that the control requests painting the {10, 10, 30, 20} area, a rectangle with the width of 20 pixels, and the height of 10 pixels, the skin will be displayed on the {14,10,26,20} as each coordinates in the "CP" syntax is added to the displayed rectangle, so the skin looks smaller. This way you can apply different effects to your objects in your control. The following screen shot shows the control's header when using a "CP:1 -6 -6 6 6", that displays the original skin on larger rectangles.

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

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

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

Use the Add method to add or replace skins to the control. The skin method, in it's simplest form, uses a single graphic file (*.ebn) assigned to a part of the control, when the "XP:" prefix is not specified in the Skin parameter (available for Windows XP systems). 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 do multiple changes to the control. Use the Refresh method to refresh the control.

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

color stored in RRGGBB format and the index of the skin (ID parameter) in the last 7 bits in the high significant byte of the color. For instance, the BackColor = BackColor Or &H2000000 indicates that we apply the skin with the index 2 using the old color, to the object that BackColor is applied.

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

- control's **borders** using the Appearance property
- **tooltip** appearance using the Background property
- panel's background using the BackColor property
- background of the panel's percent using the BackColorPercent property
- Any HTML caption that includes an <img> tag.

The following **VB** sample shows "How can I change the panel's visual appearance using EBN files":

```
With StatusBar1
   .BeginUpdate
   .Appearance = None2
   .VisualAppearance.Add 4,"c:\exontrol\images\border.ebn"
   .VisualAppearance.Add 5,"CP:4 1 1 -1 -1"
   .BackColorPanels = 83886080
   .BackColor = -2147483633
   .Format = "1,2,3,4,(5/6/7/8)"
   .Debug = True
   .EndUpdate
End With
```

The following **VB.NET** sample shows "How can I change the panel's visual appearance using EBN files":

```
With AxStatusBar1
   .BeginUpdate
   .Appearance = EXSTATUSBARLib.AppearanceEnum.None2
   .VisualAppearance.Add 4,"c:\exontrol\images\border.ebn"
   .VisualAppearance.Add 5,"CP:4 1 1 -1 -1"
   .GetOcx().BackColorPanels = &H5000000
   .GetOcx().BackColor = &H8000000f
   .Format = "1,2,3,4,(5/6/7/8)"
   .Debug = True
```

```
    .EndUpdate
End With
```

The following **C++** sample shows "How can I change the panel's visual appearance using EBN files":

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

    #import "C:\\WINNT\\system32\\ExStatusBar.dll"
    using namespace EXSTATUSBARLib;
*/
EXSTATUSBARLib::IStatusBarPtr spStatusBar1 = GetDlgItem(IDC_STATUSBAR1)-
>GetControlUnknown();
spStatusBar1->BeginUpdate();
spStatusBar1->PutAppearance(EXSTATUSBARLib::None2);
spStatusBar1->GetVisualAppearance()->Add(4,"c:\\exontrol\\images\\border.ebn");
spStatusBar1->GetVisualAppearance()->Add(5,"CP:4 1 1 -1 -1");
spStatusBar1->PutBackColorPanels(83886080);
spStatusBar1->PutBackColor(-2147483633);
spStatusBar1->PutFormat(L"1,2,3,4,(5/6/7/8)");
spStatusBar1->PutDebug(VARIANT_TRUE);
spStatusBar1->EndUpdate();
```

The following **C#** sample shows "How can I change the panel's visual appearance using EBN files":

```
axStatusBar1.BeginUpdate();
axStatusBar1.Appearance = EXSTATUSBARLib.AppearanceEnum.None2;
axStatusBar1.VisualAppearance.Add(4,"c:\\exontrol\\images\\border.ebn");
axStatusBar1.VisualAppearance.Add(5,"CP:4 1 1 -1 -1");
(axStatusBar1.GetOcx() as EXSTATUSBARLib.StatusBar).BackColorPanels = 0x5000000;
(axStatusBar1.GetOcx() as EXSTATUSBARLib.StatusBar).BackColor = 0x8000000f;
axStatusBar1.Format = "1,2,3,4,(5/6/7/8)";
axStatusBar1.Debug = true;
axStatusBar1.EndUpdate();
```

The following **VFP** sample shows "How can I change the panel's visual appearance using EBN files":

```
with thisform.StatusBar1
   .BeginUpdate
   .Appearance = 0
   .VisualAppearance.Add(4,"c:\exontrol\images\border.ebn")
   .VisualAppearance.Add(5,"CP:4 1 1 -1 -1")
   .BackColorPanels = 83886080
   .BackColor = -2147483633
   .Format = "1,2,3,4,(5/6/7/8)"
   .Debug = .T.
   .EndUpdate
endwith
```

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:

- control's **borders** using the Appearance property
- **tooltip** appearance using the Background property
- panel's background using the BackColor property
- background of the panel's percent using the BackColorPercent property
- Any HTML caption that includes an <img> tag.

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:

- control's **borders** using the Appearance property
- **tooltip** appearance using the Background property
- panel's background using the BackColor property
- background of the panel's percent using the BackColorPercent property
- Any HTML caption that includes an <img> tag.

OleEvent object

The OleEvent object holds information about an event fired by an ActiveX control hosted by a panel. Use the ControlID property to insert an ActiveX control inside a panel.

Name	Description
CountParam	Retrieves the count of the OLE event's arguments.
ID	Retrieves a long expression that specifies the identifier of the event.
Name	Retrieves the original name of the fired event.
Param	Retrieves an OleEventParam object given either the index of the parameter, or its name.
ToString	Retrieves information about the event.

property OleEvent.CountParam as Long

Retrieves the count of the OLE event's arguments.

Type	Description
Long	A long value that indicates the count of the arguments.

The CountParam property specifies the number of parameters in the event. Use the ToString property to display information about fired event.

The following VB sample enumerates the arguments of an OLE event when OLEEvent is fired.

```
Private Sub StatusBar1_OleEvent(ByVal Panel As EXSTATUSBARLibCtl.IPanel, ByVal Ev As
EXSTATUSBARLibCtl.IOleEvent)
   Debug.Print Ev.ToString()
End Sub
```

The following VC sample displays the events that an ActiveX control is firing while it is hosted by an item:

```
#import "C:\\WINNT\\SYSTEM32\\ExStatusBar.dll"
using namespace EXSTATUSBARLib;

void OnOleEventStatusbar1(LPDISPATCH Panel, LPDISPATCH Ev)
{
   EXSTATUSBARLib::IOleEventPtr spEvent( Ev );
   OutputDebugString( spEvent->ToString );
}
```

The #import clause is required to get the wrapper classes for IOleEvent and IOleEventParam objects, that are not defined by the MFC class wizard. The same #import statement defines the EXSTATUSBARLib namespace that include all objects and types of the control's TypeLibrary. In case your exstatusbar.dll library is located to another place than the system folder or well known path, the path to the library should be provided, in order to let the VC finds the type library.

The following VB.NET sample displays the events that an ActiveX control is firing while it is hosted by an item:

```
Private Sub AxStatusBar1_OleEvent(ByVal sender As System.Object, ByVal e As
```

```
AxEXSTATUSBARLib._IStatusBarEvents_OleEventEvent) Handles AxStatusBar1.OleEvent
    Debug.Print(e.ev.ToString)
End Sub
```

The following C# sample displays the events that an ActiveX control is firing while it is hosted by an item:

```
private void axStatusBar1_OleEvent(object sender,
AxEXSTATUSBARLib._IStatusBarEvents_OleEventEvent e)
{
    System.Diagnostics.Debug.Print(e.ev.ToString);
}
```

The following VFP sample displays the events that an ActiveX control fires when it is hosted by an item:

```
*** ActiveX Control Event ***
LPARAMETERS panel, ev

wait window nowait ev.ToString
```

property OleEvent.ID as Long

Retrieves a long expression that specifies the identifier of the event.

Type	Description
Long	A Long expression that defines the identifier of the OLE event.

The identifier of the event could be used to identify a specified OLE event. Use the Name property of the OLE Event to get the name of the OLE Event. Use the ToString property to display information about an OLE event. The ToString property displays the identifier of the event after the name of the event in two [] brackets. For instance, the ToString property gets the "KeyDown[-602](KeyCode/Short* = 9,Shift/Short = 0)" when TAB key is pressed, so the identifier of the KeyDown event being fired by the inside User editor is -602.

property OleEvent.Name as String

Retrieves the original name of the fired event.

Type	Description
String	A string expression that indicates the event's name.

Use the ID property to specify a specified even by its identifier. Use the ToString property to display information about fired event such us name, parameters, types and values. Use the CountParam property to count the parameters of an OLE event. Use the Param property to get the event's parameter. Use the Value property to specify the value of the parameter.

The following VB sample enumerates the arguments of an OLE event when OLEEvent is fired.

```
Private Sub StatusBar1_OleEvent(ByVal Panel As EXSTATUSBARLibCtl.IPanel, ByVal Ev As
EXSTATUSBARLibCtl.IOleEvent)
   Debug.Print "Event name:" & Ev.Name
   If (Ev.CountParam = 0) Then
      Debug.Print "The event has no arguments."
   Else
      Debug.Print "The event has the following arguments:"
      Dim i As Long
      For i = 0 To Ev.CountParam - 1
         Debug.Print Ev(i).Name; " = " & Ev(i).Value
      Next
   End If
End Sub
```

The following VC sample displays the events that an inner ActiveX control fires:

```
#import "C:\\WINNT\\SYSTEM32\\ExStatusBar.dll"
using namespace EXSTATUSBARLib;

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

```
            return szDefault;

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

void OnOleEventStatusbar1(LPDISPATCH Panel, LPDISPATCH Ev)
{
    EXSTATUSBARLib::IOleEventPtr spEvent( Ev );
    CString strOutput;
    strOutput.Format( "Event's name: %s\n", spEvent->Name.operator const char *() );
    OutputDebugString( strOutput );
    if ( spEvent->CountParam == 0 )
        OutputDebugString( "The event has no parameters." );
    else
    {
        for ( long i = 0; i < spEvent->CountParam; i++ )
        {
            EXSTATUSBARLib::IOleEventParamPtr spParam = spEvent->GetParam( COleVariant(
i ) );
            strOutput.Format( "Name: %s, Value: %s\n", spParam->Name.operator const char *
(), V2S( &spParam->Value ) );
            OutputDebugString( strOutput );
        }
    }
    OutputDebugString( "" );
}
```

The #import clause is required to get the wrapper classes for IOleEvent and IOleEventParam objects, that are not defined by the MFC class wizard. The same #import statement defines the EXSTATUSBARLib namespace that include all objects and types of the control's TypeLibrary. In case your exstatusBar.dll library is located to another place than the system folder or well known path, the path to the library should be provided, in order to let the VC finds the type library.

The following VB.NET sample displays the events that an inner ActiveX control fires:

```
Private Sub AxStatusBar1_OleEvent(ByVal sender As System.Object, ByVal e As
AxEXSTATUSBARLib._IStatusBarEvents_OleEventEvent) Handles AxStatusBar1.OleEvent
   Debug.WriteLine("Event's name: " & e.ev.Name)
   Dim i As Long
   For i = 0 To e.ev.CountParam - 1
      Dim eP As EXSTATUSBARLib.OleEventParam
      eP = e.ev(i)
      Debug.WriteLine("Name: " & e.ev.Name & " Value: " & eP.Value)
   Next
End Sub
```

The following C# sample displays the events that an inner ActiveX control fires:

```
private void axStatusBar1_OleEvent(object sender,
AxEXSTATUSBARLib._IStatusBarEvents_OleEventEvent e)
{
   System.Diagnostics.Debug.WriteLine("Event's name: " + e.ev.Name.ToString());
   for (int i = 0; i < e.ev.CountParam; i++)
   {
      EXSTATUSBARLib.IOleEventParam evP = e.ev[i];
      System.Diagnostics.Debug.WriteLine("Name: " + evP.Name.ToString() + ", Value: " +
evP.Value.ToString());
   }
}
```

The following VFP sample displays the events that an inner ActiveX control fires:

```
*** ActiveX Control Event ***
LPARAMETERS panel, ev

local s
s = "Event's name: " + ev.Name
for i = 0 to ev.CountParam - 1
      s = s +  "Name: " + ev.Param(i).Name + " ,Value: " + Str(ev.Param(i).Value)
endfor
wait window nowait s
```

property OleEvent.Param (item as Variant) as OleEventParam

Retrieves an OleEventParam object given either the index of the parameter, or its name.

Type	Description
item as Variant	A long expression that indicates the argument's index or a string expression that indicates the argument's name.
OleEventParam	An OleEventParam object that contains the name and the value for the argument.

Use the Param property to get the event's parameter. Use the ID property to specify a specified even by its identifier. Use the ToString property to display information about fired event such us name, parameters, types and values. Use the CountParam property to count the parameters of an OLE event. Use the Value property to specify the value of the parameter.

The following VB sample enumerates the arguments of an OLE event when OLEEvent is fired.

```
Private Sub StatusBar1_OleEvent(ByVal Panel As EXSTATUSBARLibCtl.IPanel, ByVal Ev As
EXSTATUSBARLibCtl.IOleEvent)
   Debug.Print "Event name:" & Ev.Name
   If (Ev.CountParam = 0) Then
      Debug.Print "The event has no arguments."
   Else
      Debug.Print "The event has the following arguments:"
      Dim i As Long
      For i = 0 To Ev.CountParam - 1
         Debug.Print Ev(i).Name; " = " & Ev(i).Value
      Next
   End If
End Sub
```

The following VC sample displays the events that an inner ActiveX control fires:

```
#import "C:\\WINNT\\SYSTEM32\\ExStatusBar.dll"
using namespace EXSTATUSBARLib;

static CString V2S( VARIANT* pv, LPCTSTR szDefault = _T("") )
{
```

```
    if ( pv )
    {
        if ( pv->vt == VT_ERROR )
            return szDefault;

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

void OnOleEventStatusbar1(LPDISPATCH Panel, LPDISPATCH Ev)
{
    EXSTATUSBARLib::IOleEventPtr spEvent( Ev );
    CString strOutput;
    strOutput.Format( "Event's name: %s\n", spEvent->Name.operator const char *() );
    OutputDebugString( strOutput );
    if ( spEvent->CountParam == 0 )
        OutputDebugString( "The event has no parameters." );
    else
    {
        for ( long i = 0; i < spEvent->CountParam; i++ )
        {
            EXSTATUSBARLib::IOleEventParamPtr spParam = spEvent->GetParam( COleVariant(
i ) );
            strOutput.Format( "Name: %s, Value: %s\n", spParam->Name.operator const char *
(), V2S( &spParam->Value ) );
            OutputDebugString( strOutput );
        }
    }
    OutputDebugString( "" );
}
```

The #import clause is required to get the wrapper classes for IOleEvent and IOleEventParam objects, that are not defined by the MFC class wizard. The same #import statement defines the EXSTATUSBARLib namespace that include all objects and types of the control's TypeLibrary. In case your exstatusBar.dll library is located to another place

than the system folder or well known path, the path to the library should be provided, in order to let the VC finds the type library.

The following VB.NET sample displays the events that an inner ActiveX control fires:

```
Private Sub AxStatusBar1_OleEvent(ByVal sender As System.Object, ByVal e As
AxEXSTATUSBARLib._IStatusBarEvents_OleEventEvent) Handles AxStatusBar1.OleEvent
   Debug.WriteLine("Event's name: " & e.ev.Name)
   Dim i As Long
   For i = 0 To e.ev.CountParam - 1
      Dim eP As EXSTATUSBARLib.OleEventParam
      eP = e.ev(i)
      Debug.WriteLine("Name: " & e.ev.Name & " Value: " & eP.Value)
   Next
End Sub
```

The following C# sample displays the events that an inner ActiveX control fires:

```
private void axStatusBar1_OleEvent(object sender,
AxEXSTATUSBARLib._IStatusBarEvents_OleEventEvent e)
{
   System.Diagnostics.Debug.WriteLine("Event's name: " + e.ev.Name.ToString());
   for (int i = 0; i < e.ev.CountParam; i++)
   {
      EXSTATUSBARLib.IOleEventParam evP = e.ev[i];
      System.Diagnostics.Debug.WriteLine("Name: " + evP.Name.ToString() + ", Value: " +
evP.Value.ToString());
   }
}
```

The following VFP sample displays the events that an inner ActiveX control fires:

```
*** ActiveX Control Event ***
LPARAMETERS panel, ev

local s
s = "Event's name: " + ev.Name
for i = 0 to ev.CountParam - 1
      s = s +  "Name: " + ev.Param(i).Name + " ,Value: " + Str(ev.Param(i).Value)
```

```
endfor
wait window nowait s
```

property OleEvent.ToString as String

Retrieves information about the event.

Type	Description
String	A String expression that shows information about an OLE event. The ToString property gets the information as follows: Name[ID] (Param/Type = Value, Param/Type = Value, ...). For instance, "KeyDown[-602] (KeyCode/Short* = 9,Shift/Short = 0)" indicates that the KeyDown event is fired, with the identifier -602 with two parameters KeyCode as a reference to a short type with the value 8, and Shift parameter as Short type with the value 0.

Use the ToString property to display information about fired event such us name, parameters, types and values. Using the ToString property you can quickly identifies the event that you should handle in your application. Use the ID property to specify a specified even by its identifier. Use the Name property to get the name of the event. Use the Param property to access a specified parameter using its index or its name.

Displaying ToString property during the OLE Event event may show data like follows:

```
MouseMove[-606](Button/Short = 0,Shift/Short = 0,X/Long = 46,Y/Long = 15)
MouseDown[-605](Button/Short = 1,Shift/Short = 0,X/Long = 46,Y/Long = 15)
KeyDown[-602](KeyCode/Short* = 83,Shift/Short = 0)
KeyPress[-603](KeyAscii/Short* = 115)
Change[2]()
KeyUp[-604](KeyCode/Short* = 83,Shift/Short = 0)
MouseUp[-607](Button/Short = 1,Shift/Short = 0,X/Long = 46,Y/Long = 15)
MouseMove[-606](Button/Short = 0,Shift/Short = 0,X/Long = 46,Y/Long = 15)
```

OleEventParam object

The OleEventParam holds the name and the value for an event's argument.

Name	Description
Name	Retrieves the name of the event's parameter.
Value	Retrieves the value of the event's parameter.

property OleEventParam.Name as String

Retrieves the name of the event's parameter.

Type	Description
String	A string expression that indicates the name of the event's parameter.

The following VB sample enumerates the arguments of an OLE event when OLEEvent is fired.

```
Private Sub StatusBar1_OleEvent(ByVal Panel As EXSTATUSBARLibCtl.IPanel, ByVal Ev As
EXSTATUSBARLibCtl.IOleEvent)
   Debug.Print "Event name:" & Ev.Name
   If (Ev.CountParam = 0) Then
      Debug.Print "The event has no arguments."
   Else
      Debug.Print "The event has the following arguments:"
      Dim i As Long
      For i = 0 To Ev.CountParam - 1
         Debug.Print Ev(i).Name; " = " & Ev(i).Value
      Next
   End If
End Sub
```

The following VC sample displays the events that an inner ActiveX control fires:

```
#import "C:\\WINNT\\SYSTEM32\\ExStatusBar.dll"
using namespace EXSTATUSBARLib;

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

      COleVariant vt;
      vt.ChangeType( VT_BSTR, pv );
```

```
        return V_BSTR( &vt );
    }
    return szDefault;
}

void OnOleEventStatusbar1(LPDISPATCH Panel, LPDISPATCH Ev)
{
    EXSTATUSBARLib::IOleEventPtr spEvent( Ev );
    CString strOutput;
    strOutput.Format( "Event's name: %s\n", spEvent->Name.operator const char *() );
    OutputDebugString( strOutput );
    if ( spEvent->CountParam == 0 )
        OutputDebugString( "The event has no parameters." );
    else
    {
        for ( long i = 0; i < spEvent->CountParam; i++ )
        {
            EXSTATUSBARLib::IOleEventParamPtr spParam = spEvent->GetParam( COleVariant(
i ) );
            strOutput.Format( "Name: %s, Value: %s\n", spParam->Name.operator const char *
(), V2S( &spParam->Value ) );
            OutputDebugString( strOutput );
        }
    }
    OutputDebugString( "" );
}
```

The #import clause is required to get the wrapper classes for IOleEvent and IOleEventParam objects, that are not defined by the MFC class wizard. The same #import statement defines the EXSTATUSBARLib namespace that include all objects and types of the control's TypeLibrary. In case your exstatusBar.dll library is located to another place than the system folder or well known path, the path to the library should be provided, in order to let the VC finds the type library.

The following VB.NET sample displays the events that an inner ActiveX control fires:

```
Private Sub AxStatusBar1_OleEvent(ByVal sender As System.Object, ByVal e As
AxEXSTATUSBARLib._IStatusBarEvents_OleEventEvent) Handles AxStatusBar1.OleEvent
    Debug.WriteLine("Event's name: " & e.ev.Name)
```

```
    Dim i As Long
    For i = 0 To e.ev.CountParam - 1
        Dim eP As EXSTATUSBARLib.OleEventParam
        eP = e.ev(i)
        Debug.WriteLine("Name: " & e.ev.Name & " Value: " & eP.Value)
    Next
End Sub
```

The following C# sample displays the events that an inner ActiveX control fires:

```
private void axStatusBar1_OleEvent(object sender,
AxEXSTATUSBARLib._IStatusBarEvents_OleEventEvent e)
{
    System.Diagnostics.Debug.WriteLine("Event's name: " + e.ev.Name.ToString());
    for (int i = 0; i < e.ev.CountParam; i++)
    {
        EXSTATUSBARLib.IOleEventParam evP = e.ev[i];
        System.Diagnostics.Debug.WriteLine("Name: " + evP.Name.ToString() + ", Value: " +
evP.Value.ToString());
    }
}
```

The following VFP sample displays the events that an inner ActiveX control fires:

```
*** ActiveX Control Event ***
LPARAMETERS panel, ev

local s
s = "Event's name: " + ev.Name
for i = 0 to ev.CountParam - 1
        s = s +  "Name: " + ev.Param(i).Name + " ,Value: " + Str(ev.Param(i).Value)
endfor
wait window nowait s
```

property OleEventParam.Value as Variant

Retrieves or sets the value of the event's parameter.

Type	Description
Variant	A variant value that indicates the value of the event's parameter.

Use the Value property to specify the value of the parameter. Use the ID property to specify a specified even by its identifier. Use the ToString property to display information about fired event such us name, parameters, types and values. Use the CountParam property to count the parameters of an OLE event. Use the Param property to get the event's parameter.

The following VB sample enumerates the arguments of an OLE event when OLEEvent is fired.

```
Private Sub StatusBar1_OleEvent(ByVal Panel As EXSTATUSBARLibCtl.IPanel, ByVal Ev As
EXSTATUSBARLibCtl.IOleEvent)
   Debug.Print "Event name:" & Ev.Name
   If (Ev.CountParam = 0) Then
      Debug.Print "The event has no arguments."
   Else
      Debug.Print "The event has the following arguments:"
      Dim i As Long
      For i = 0 To Ev.CountParam - 1
         Debug.Print Ev(i).Name; " = " & Ev(i).Value
      Next
   End If
End Sub
```

The following VC sample displays the events that an inner ActiveX control fires:

```
#import "C:\\WINNT\\SYSTEM32\\ExStatusBar.dll"
using namespace EXSTATUSBARLib;

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

```
            return szDefault;

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

void OnOleEventStatusbar1(LPDISPATCH Panel, LPDISPATCH Ev)
{
    EXSTATUSBARLib::IOleEventPtr spEvent( Ev );
    CString strOutput;
    strOutput.Format( "Event's name: %s\n", spEvent->Name.operator const char *() );
    OutputDebugString( strOutput );
    if ( spEvent->CountParam == 0 )
        OutputDebugString( "The event has no parameters." );
    else
    {
        for ( long i = 0; i < spEvent->CountParam; i++ )
        {
            EXSTATUSBARLib::IOleEventParamPtr spParam = spEvent->GetParam( COleVariant(
i ) );
            strOutput.Format( "Name: %s, Value: %s\n", spParam->Name.operator const char *
(), V2S( &spParam->Value ) );
            OutputDebugString( strOutput );
        }
    }
    OutputDebugString( "" );
}
```

The #import clause is required to get the wrapper classes for IOleEvent and IOleEventParam objects, that are not defined by the MFC class wizard. The same #import statement defines the EXSTATUSBARLib namespace that include all objects and types of the control's TypeLibrary. In case your exstatusBar.dll library is located to another place than the system folder or well known path, the path to the library should be provided, in order to let the VC finds the type library.

The following VB.NET sample displays the events that an inner ActiveX control fires:

```
Private Sub AxStatusBar1_OleEvent(ByVal sender As System.Object, ByVal e As
AxEXSTATUSBARLib._IStatusBarEvents_OleEventEvent) Handles AxStatusBar1.OleEvent
   Debug.WriteLine("Event's name: " & e.ev.Name)
   Dim i As Long
   For i = 0 To e.ev.CountParam - 1
      Dim eP As EXSTATUSBARLib.OleEventParam
      eP = e.ev(i)
      Debug.WriteLine("Name: " & e.ev.Name & " Value: " & eP.Value)
   Next
End Sub
```

The following C# sample displays the events that an inner ActiveX control fires:

```
private void axStatusBar1_OleEvent(object sender,
AxEXSTATUSBARLib._IStatusBarEvents_OleEventEvent e)
{
   System.Diagnostics.Debug.WriteLine("Event's name: " + e.ev.Name.ToString());
   for (int i = 0; i < e.ev.CountParam; i++)
   {
      EXSTATUSBARLib.IOleEventParam evP = e.ev[i];
      System.Diagnostics.Debug.WriteLine("Name: " + evP.Name.ToString() + ", Value: " +
evP.Value.ToString());
   }
}
```

The following VFP sample displays the events that an inner ActiveX control fires:

```
*** ActiveX Control Event ***
LPARAMETERS panel, ev

local s
s = "Event's name: " + ev.Name
for i = 0 to ev.CountParam - 1
      s = s +  "Name: " + ev.Param(i).Name + " ,Value: " + Str(ev.Param(i).Value)
endfor
wait window nowait s
```

Panel object

The Panel object identifies a panel area in the status bar control. The Panel object supports the following properties and methods:

Name	Description
Alignment	Gets or sets the alignment of text and icons within the status bar panel.
BackColor	Specifies the background color or the visual appearance of the panel.
BackColorPercent	Specifies the background color or the visual appearance of the percent bar in the panel.
BackgroundExt	Indicates additional colors, text, images that can be displayed on the panel's background using the EBN string format.
BackgroundExtValue	Specifies at runtime, the value of the giving property for specified part of the background extension.
Bold	Specifies whether the text in the panel appears in bold.
ControlID	Specifies the program identifier being shown in the panel.
Enabled	Specifies whether the panel is enabled or disabled.
ForeColor	Specifies the foreground color of the text in the panel.
Height	Specifies the height in pixels of the panel.
Image	Gets or sets the index of the icon to display within the status bar panel.
Index	Retrieves the identifier of the panel in the status bar.
Italic	Specifies whether the text in the panel appears in italic.
License	Specifies the runtime license required to create the user control inside the panel.
Object	Retrieves the inside control being created by ControlID property.
Offset	Specifies the offset to apply when text is being diplayed.
OffsetPercent	Specifies the offset to apply when the percent bar is displayed on the panel.
OwnerDraw	Specifies whether the user is responsible with painting the panel in the status bar control.
Percent	Specifies the percent to display the background.

StrikeOut	Specifies whether the text in the panel appears as strikeout.
Text	Gets or sets the text of the status bar panel.
ToolTipText	Gets or sets ToolTip text associated with the status bar panel.
ToolTipTitle	Gets or sets ToolTip title associated with the status bar panel.
Transparency	Specifies the transparency to display the text in the panel.
Underline	Specifies whether the text in the panel appears as underlined.
UserData	Associates an extra data to the panel.
Width	Specifies the width in pixels of the panel.
WordWrap	Specifies whether the text is word wrapping in the panel.

property Panel.Alignment as TextAlignEnum

Gets or sets the alignment of text and icons within the status bar panel.

Type	Description
TextAlignEnum	A TextAlignEnum that specifies the alignment of the caption in the panel.

Use the Alignment property to align the caption in the panel. Use the Text property to assign a caption to a panel. Use the Image property to assign an icon to a panel.

The following **VB** sample shows "How can I align the text inside the panel":

```
With StatusBar1
  .BeginUpdate
  .VisualAppearance.Add 4,"c:\exontrol\images\border.ebn"
  .VisualAppearance.Add 5,"CP:4 1 1 -1 -1"
  .BackColorPanels = 83886080
  .BackColor = -2147483633
  .Format = "(0/16/32),(1/17/33),(2/18/34)"
  With .Panel(0)
    .Text = "exAlignTopLeft"
    .Alignment = exAlignTopLeft
  End With
  With .Panel(1)
    .Text = "exAlignTopCenter"
    .Alignment = exAlignTopCenter
  End With
  With .Panel(2)
    .Text = "exAlignTopRight"
    .Alignment = exAlignTopRight
  End With
  With .Panel(16)
    .Text = "exAlignMiddleLeft"
    .Alignment = exAlignMiddleLeft
  End With
  With .Panel(17)
    .Text = "exAlignMiddleCenter"
    .Alignment = exAlignMiddleCenter
```

```
    End With
    With .Panel(18)
        .Text = "exAlignMiddleRight"
        .Alignment = exAlignMiddleRight
    End With
    With .Panel(32)
        .Text = "exAlignBottomLeft"
        .Alignment = exAlignBottomLeft
    End With
    With .Panel(33)
        .Text = "exAlignBottomCenter"
        .Alignment = exAlignBottomCenter
    End With
    With .Panel(34)
        .Text = "exAlignBottomRight"
        .Alignment = exAlignBottomRight
    End With
    .EndUpdate
End With
```

The following **VB.NET** sample shows "How can I align the text inside the panel":

```
With AxStatusBar1
    .BeginUpdate
    .VisualAppearance.Add 4,"c:\exontrol\images\border.ebn"
    .VisualAppearance.Add 5,"CP:4 1 1 -1 -1"
    .GetOcx().BackColorPanels = &H5000000
    .GetOcx().BackColor = &H8000000f
    .Format = "(0/16/32),(1/17/33),(2/18/34)"
    With .get_Panel(0)
        .Text = "exAlignTopLeft"
        .Alignment = EXSTATUSBARLib.TextAlignEnum.exAlignTopLeft
    End With
    With .get_Panel(1)
        .Text = "exAlignTopCenter"
        .Alignment = EXSTATUSBARLib.TextAlignEnum.exAlignTopCenter
    End With
```

```
    With .get_Panel(2)
        .Text = "exAlignTopRight"
        .Alignment = EXSTATUSBARLib.TextAlignEnum.exAlignTopRight
    End With
    With .get_Panel(16)
        .Text = "exAlignMiddleLeft"
        .Alignment = EXSTATUSBARLib.TextAlignEnum.exAlignMiddleLeft
    End With
    With .get_Panel(17)
        .Text = "exAlignMiddleCenter"
        .Alignment = EXSTATUSBARLib.TextAlignEnum.exAlignMiddleCenter
    End With
    With .get_Panel(18)
        .Text = "exAlignMiddleRight"
        .Alignment = EXSTATUSBARLib.TextAlignEnum.exAlignMiddleRight
    End With
    With .get_Panel(32)
        .Text = "exAlignBottomLeft"
        .Alignment = EXSTATUSBARLib.TextAlignEnum.exAlignBottomLeft
    End With
    With .get_Panel(33)
        .Text = "exAlignBottomCenter"
        .Alignment = EXSTATUSBARLib.TextAlignEnum.exAlignBottomCenter
    End With
    With .get_Panel(34)
        .Text = "exAlignBottomRight"
        .Alignment = EXSTATUSBARLib.TextAlignEnum.exAlignBottomRight
    End With
    .EndUpdate
End With
```

The following **C++** sample shows "How can I align the text inside the panel":

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

```
   #import "C:\\WINNT\\system32\\ExStatusBar.dll"
   using namespace EXSTATUSBARLib;
*/
EXSTATUSBARLib::IStatusBarPtr spStatusBar1 = GetDlgItem(IDC_STATUSBAR1)-
>GetControlUnknown();
spStatusBar1->BeginUpdate();
spStatusBar1->GetVisualAppearance()->Add(4,"c:\\exontrol\\images\\border.ebn");
spStatusBar1->GetVisualAppearance()->Add(5,"CP:4 1 1 -1 -1");
spStatusBar1->PutBackColorPanels(83886080);
spStatusBar1->PutBackColor(-2147483633);
spStatusBar1->PutFormat(L"(0/16/32),(1/17/33),(2/18/34)");
EXSTATUSBARLib::IPanelPtr var_Panel = spStatusBar1->GetPanel(long(0));
   var_Panel->PutText(L"exAlignTopLeft");
   var_Panel->PutAlignment(EXSTATUSBARLib::exAlignTopLeft);
EXSTATUSBARLib::IPanelPtr var_Panel1 = spStatusBar1->GetPanel(long(1));
   var_Panel1->PutText(L"exAlignTopCenter");
   var_Panel1->PutAlignment(EXSTATUSBARLib::exAlignTopCenter);
EXSTATUSBARLib::IPanelPtr var_Panel2 = spStatusBar1->GetPanel(long(2));
   var_Panel2->PutText(L"exAlignTopRight");
   var_Panel2->PutAlignment(EXSTATUSBARLib::exAlignTopRight);
EXSTATUSBARLib::IPanelPtr var_Panel3 = spStatusBar1->GetPanel(long(16));
   var_Panel3->PutText(L"exAlignMiddleLeft");
   var_Panel3->PutAlignment(EXSTATUSBARLib::exAlignMiddleLeft);
EXSTATUSBARLib::IPanelPtr var_Panel4 = spStatusBar1->GetPanel(long(17));
   var_Panel4->PutText(L"exAlignMiddleCenter");
   var_Panel4->PutAlignment(EXSTATUSBARLib::exAlignMiddleCenter);
EXSTATUSBARLib::IPanelPtr var_Panel5 = spStatusBar1->GetPanel(long(18));
   var_Panel5->PutText(L"exAlignMiddleRight");
   var_Panel5->PutAlignment(EXSTATUSBARLib::exAlignMiddleRight);
EXSTATUSBARLib::IPanelPtr var_Panel6 = spStatusBar1->GetPanel(long(32));
   var_Panel6->PutText(L"exAlignBottomLeft");
   var_Panel6->PutAlignment(EXSTATUSBARLib::exAlignBottomLeft);
EXSTATUSBARLib::IPanelPtr var_Panel7 = spStatusBar1->GetPanel(long(33));
   var_Panel7->PutText(L"exAlignBottomCenter");
   var_Panel7->PutAlignment(EXSTATUSBARLib::exAlignBottomCenter);
EXSTATUSBARLib::IPanelPtr var_Panel8 = spStatusBar1->GetPanel(long(34));
```

```
   var_Panel8->PutText(L"exAlignBottomRight");
   var_Panel8->PutAlignment(EXSTATUSBARLib::exAlignBottomRight);
spStatusBar1->EndUpdate();
```

The following **C#** sample shows "How can I align the text inside the panel":

```
axStatusBar1.BeginUpdate();
axStatusBar1.VisualAppearance.Add(4,"c:\\exontrol\\images\\border.ebn");
axStatusBar1.VisualAppearance.Add(5,"CP:4 1 1 -1 -1");
(axStatusBar1.GetOcx() as EXSTATUSBARLib.StatusBar).BackColorPanels = 0x5000000;
(axStatusBar1.GetOcx() as EXSTATUSBARLib.StatusBar).BackColor = 0x8000000f;
axStatusBar1.Format = "(0/16/32),(1/17/33),(2/18/34)";
EXSTATUSBARLib.Panel var_Panel = axStatusBar1.get_Panel(0);
   var_Panel.Text = "exAlignTopLeft";
   var_Panel.Alignment = EXSTATUSBARLib.TextAlignEnum.exAlignTopLeft;
EXSTATUSBARLib.Panel var_Panel1 = axStatusBar1.get_Panel(1);
   var_Panel1.Text = "exAlignTopCenter";
   var_Panel1.Alignment = EXSTATUSBARLib.TextAlignEnum.exAlignTopCenter;
EXSTATUSBARLib.Panel var_Panel2 = axStatusBar1.get_Panel(2);
   var_Panel2.Text = "exAlignTopRight";
   var_Panel2.Alignment = EXSTATUSBARLib.TextAlignEnum.exAlignTopRight;
EXSTATUSBARLib.Panel var_Panel3 = axStatusBar1.get_Panel(16);
   var_Panel3.Text = "exAlignMiddleLeft";
   var_Panel3.Alignment = EXSTATUSBARLib.TextAlignEnum.exAlignMiddleLeft;
EXSTATUSBARLib.Panel var_Panel4 = axStatusBar1.get_Panel(17);
   var_Panel4.Text = "exAlignMiddleCenter";
   var_Panel4.Alignment = EXSTATUSBARLib.TextAlignEnum.exAlignMiddleCenter;
EXSTATUSBARLib.Panel var_Panel5 = axStatusBar1.get_Panel(18);
   var_Panel5.Text = "exAlignMiddleRight";
   var_Panel5.Alignment = EXSTATUSBARLib.TextAlignEnum.exAlignMiddleRight;
EXSTATUSBARLib.Panel var_Panel6 = axStatusBar1.get_Panel(32);
   var_Panel6.Text = "exAlignBottomLeft";
   var_Panel6.Alignment = EXSTATUSBARLib.TextAlignEnum.exAlignBottomLeft;
EXSTATUSBARLib.Panel var_Panel7 = axStatusBar1.get_Panel(33);
   var_Panel7.Text = "exAlignBottomCenter";
   var_Panel7.Alignment = EXSTATUSBARLib.TextAlignEnum.exAlignBottomCenter;
EXSTATUSBARLib.Panel var_Panel8 = axStatusBar1.get_Panel(34);
```

```
    var_Panel8.Text = "exAlignBottomRight";
    var_Panel8.Alignment = EXSTATUSBARLib.TextAlignEnum.exAlignBottomRight;
axStatusBar1.EndUpdate();
```

The following **VFP** sample shows "How can I align the text inside the panel":

```
with thisform.StatusBar1
    .BeginUpdate
    .VisualAppearance.Add(4,"c:\exontrol\images\border.ebn")
    .VisualAppearance.Add(5,"CP:4 1 1 -1 -1")
    .BackColorPanels = 83886080
    .BackColor = -2147483633
    .Format = "(0/16/32),(1/17/33),(2/18/34)"
    with .Panel(0)
        .Text = "exAlignTopLeft"
        .Alignment = 0
    endwith
    with .Panel(1)
        .Text = "exAlignTopCenter"
        .Alignment = 1
    endwith
    with .Panel(2)
        .Text = "exAlignTopRight"
        .Alignment = 2
    endwith
    with .Panel(16)
        .Text = "exAlignMiddleLeft"
        .Alignment = 16
    endwith
    with .Panel(17)
        .Text = "exAlignMiddleCenter"
        .Alignment = 17
    endwith
    with .Panel(18)
        .Text = "exAlignMiddleRight"
        .Alignment = 18
    endwith
```

```
    with .Panel(32)
        .Text = "exAlignBottomLeft"
        .Alignment = 32
    endwith
    with .Panel(33)
        .Text = "exAlignBottomCenter"
        .Alignment = 33
    endwith
    with .Panel(34)
        .Text = "exAlignBottomRight"
        .Alignment = 34
    endwith
    .EndUpdate
endwith
```

property Panel.BackColor as Color

Specifies the background color or the visual appearance of the panel.

Type	Description
Color	A Color expression that specifies the panel'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 property to change the visual appearance of the the panel using EBN files. Use the BackColorPanels property to assign the same visual appearance to all panels in the status bar. Use the <bgcolor> property to specify a background color for a portion of the caption in the panel. Use the BackColorPercent property to change the visual appearance of the progress bar in the panel. Use the ForeColor property to change the panel's foreground color.

The following **VB** sample shows "How can I change the panel's background color":

```
With StatusBar1
   .BeginUpdate
   .Format = """":4,((4;""/1/4;""),""":4,(4;""/2/4;"")),""":4"
   With .Panel(1)
      .Text = "Panel 1"
      .BackColor = 65535
   End With
   With .Panel(2)
      .Text = "Panel 2"
      .BackColor = 16711935
   End With
   .EndUpdate
End With
```

The following **VB.NET** sample shows "How can I change the panel's background color":

```
With AxStatusBar1
   .BeginUpdate
   .Format = """":4,((4;""/1/4;""),""":4,(4;""/2/4;"")),""":4"
```

```
    With .get_Panel(1)
        .Text = "Panel 1"
        .BackColor = 65535
    End With
    With .get_Panel(2)
        .Text = "Panel 2"
        .BackColor = 16711935
    End With
    .EndUpdate
End With
```

The following **C++** sample shows "How can I change the panel's background color":

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

    #import "C:\\WINNT\\system32\\ExStatusBar.dll"
    using namespace EXSTATUSBARLib;
*/
EXSTATUSBARLib::IStatusBarPtr spStatusBar1 = GetDlgItem(IDC_STATUSBAR1)-
>GetControlUnknown();
spStatusBar1->BeginUpdate();
spStatusBar1->PutFormat(L"\"\":4,((4;\"\"/1/4;\"\"),\"\":4,(4;\"\"/2/4;\"\")),\"\":4");
EXSTATUSBARLib::IPanelPtr var_Panel = spStatusBar1->GetPanel(long(1));
    var_Panel->PutText(L"Panel 1");
    var_Panel->PutBackColor(65535);
EXSTATUSBARLib::IPanelPtr var_Panel1 = spStatusBar1->GetPanel(long(2));
    var_Panel1->PutText(L"Panel 2");
    var_Panel1->PutBackColor(16711935);
spStatusBar1->EndUpdate();
```

The following **C#** sample shows "How can I change the panel's background color":

```
axStatusBar1.BeginUpdate();
axStatusBar1.Format = "\"\":4,((4;\"\"/1/4;\"\"),\"\":4,(4;\"\"/2/4;\"\")),\"\":4";
EXSTATUSBARLib.Panel var_Panel = axStatusBar1.get_Panel(1);
```

```
   var_Panel.Text = "Panel 1";
   var_Panel.BackColor = 65535;
EXSTATUSBARLib.Panel var_Panel1 = axStatusBar1.get_Panel(2);
   var_Panel1.Text = "Panel 2";
   var_Panel1.BackColor = 16711935;
axStatusBar1.EndUpdate();
```

The following **VFP** sample shows "How can I change the panel's background color":

```
with thisform.StatusBar1
   .BeginUpdate
   .Format = ""+chr(34)+""+chr(34)+":4,
((4;"+chr(34)+""+chr(34)+"/1/4;"+chr(34)+""+chr(34)+"),"+chr(34)+""+chr(34)+":4,
(4;"+chr(34)+""+chr(34)+"/2/4;"+chr(34)+""+chr(34)+")),"+chr(34)+""+chr(34)+":4"
   with .Panel(1)
      .Text = "Panel 1"
      .BackColor = 65535
   endwith
   with .Panel(2)
      .Text = "Panel 2"
      .BackColor = 16711935
   endwith
   .EndUpdate
endwith
```

property Panel.BackColorPercent as Color

Specifies the background color or the visual appearance of the percent bar in the panel.

Type	Description
Color	A Color expression that specifies the progress bar '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 BackColorPercent property to change the visual appearance of the progress bar in the panel. Use the Percent value to display a progress bar inside the panel. Use the BackColor property to change the visual appearance of the the panel using EBN files. Use the BackColorPanels property to assign the same visual appearance to all panels in the status bar. Use the ForeColor property to change the panel's foreground color.

The following **VB** sample shows "How can I change the color of the percent or a progress-bar inside the panel":

```
With StatusBar1
   .BeginUpdate
   .VisualAppearance.Add 4,"c:\exontrol\images\border.ebn"
   .VisualAppearance.Add 5,"CP:4 1 1 -1 -1"
   .BackColorPanels = 83886080
   .BackColor = -2147483633
   .Format = "1/2,(24;5/6)"
   With .Panel(5)
      .Text = "15%"
      .Percent = 15
      .BackColorPercent = 255
   End With
   .EndUpdate
End With
```

The following **VB.NET** sample shows "How can I change the color of the percent or a progress-bar inside the panel":

```
With AxStatusBar1
   .BeginUpdate
   .VisualAppearance.Add 4,"c:\exontrol\images\border.ebn"
   .VisualAppearance.Add 5,"CP:4 1 1 -1 -1"
   .GetOcx().BackColorPanels = &H5000000
   .GetOcx().BackColor = &H8000000f
   .Format = "1/2,(24;5/6)"
   With .get_Panel(5)
      .Text = "15%"
      .Percent = 15
      .BackColorPercent = 255
   End With
   .EndUpdate
End With
```

The following **C++** sample shows "How can I change the color of the percent or a progress-bar inside the panel":

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

   #import "C:\\WINNT\\system32\\ExStatusBar.dll"
   using namespace EXSTATUSBARLib;
*/
EXSTATUSBARLib::IStatusBarPtr spStatusBar1 = GetDlgItem(IDC_STATUSBAR1)-
>GetControlUnknown();
spStatusBar1->BeginUpdate();
spStatusBar1->GetVisualAppearance()->Add(4,"c:\\exontrol\\images\\border.ebn");
spStatusBar1->GetVisualAppearance()->Add(5,"CP:4 1 1 -1 -1");
spStatusBar1->PutBackColorPanels(83886080);
spStatusBar1->PutBackColor(-2147483633);
spStatusBar1->PutFormat(L"1/2,(24;5/6)");
EXSTATUSBARLib::IPanelPtr var_Panel = spStatusBar1->GetPanel(long(5));
   var_Panel->PutText(L"15%");
   var_Panel->PutPercent(15);
```

```
    var_Panel->PutBackColorPercent(255);
spStatusBar1->EndUpdate();
```

The following **C#** sample shows "How can I change the color of the percent or a progress-bar inside the panel":

```
axStatusBar1.BeginUpdate();
axStatusBar1.VisualAppearance.Add(4,"c:\\exontrol\\images\\border.ebn");
axStatusBar1.VisualAppearance.Add(5,"CP:4 1 1 -1 -1");
(axStatusBar1.GetOcx() as EXSTATUSBARLib.StatusBar).BackColorPanels = 0x5000000;
(axStatusBar1.GetOcx() as EXSTATUSBARLib.StatusBar).BackColor = 0x8000000f;
axStatusBar1.Format = "1/2,(24;5/6)";
EXSTATUSBARLib.Panel var_Panel = axStatusBar1.get_Panel(5);
    var_Panel.Text = "15%";
    var_Panel.Percent = 15;
    var_Panel.BackColorPercent = 255;
axStatusBar1.EndUpdate();
```

The following **VFP** sample shows "How can I change the color of the percent or a progress-bar inside the panel":

```
with thisform.StatusBar1
    .BeginUpdate
    .VisualAppearance.Add(4,"c:\exontrol\images\border.ebn")
    .VisualAppearance.Add(5,"CP:4 1 1 -1 -1")
    .BackColorPanels = 83886080
    .BackColor = -2147483633
    .Format = "1/2,(24;5/6)"
    with .Panel(5)
        .Text = "15%"
        .Percent = 15
        .BackColorPercent = 255
    endwith
    .EndUpdate
endwith
```

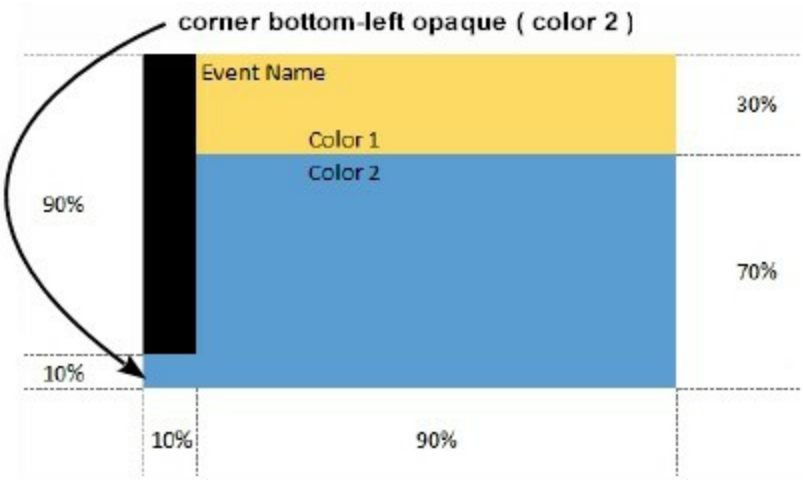
property Panel.BackgroundExt as String

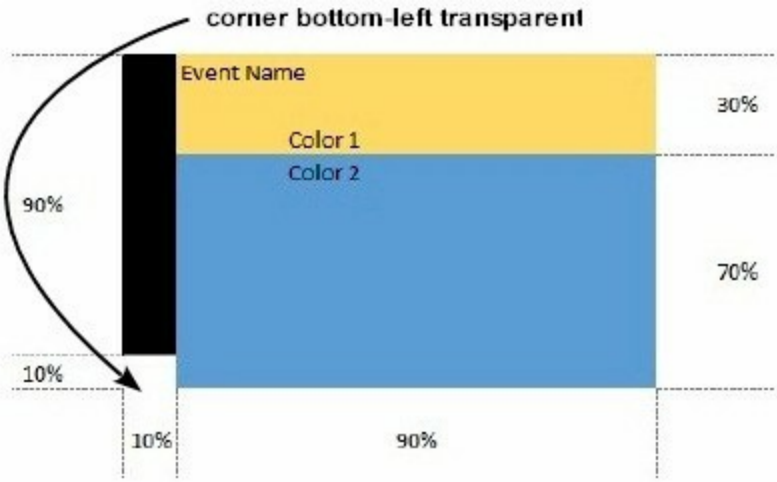
Indicates additional colors, text, images that can be displayed on the object's background using the EBN string format.

Type	Description
String	A String expression ("**EBN String Format**") that defines the layout of the UI to be applied on the object's background. The syntax of EBN String Format in BNF notation is shown bellow. *You can use the EBN's Builder of eXButton/COM control to define visually the EBN String Format.*

By default, the BackgroundExt property is empty. Using the BackgroundExt property you have unlimited options to show any HTML text, images, colors, EBNs, patterns, frames anywhere on the object's background. *For instance, let's say you need to display **more** colors on the object's background, or just want to display an **additional** caption or image to a specified location on the object's background.* The EBN String Format defines the parts of the EBN to be applied on the object's background. The EBN is a set of UI elements that are built as a tree where each element is anchored to its parent element. Use the BackgroundExtValue property to change at runtime any UI property for any part that composes the EBN String Format. The BackgroundExt property is applied right after setting the object's backcolor, and before drawing the default object's captions, icons or pictures.

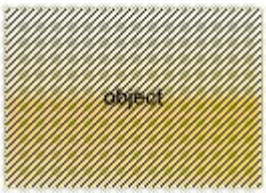
Complex samples:



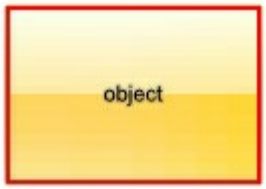


Easy samples:

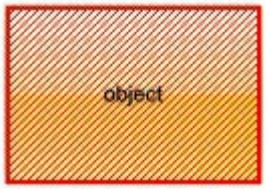
- "[pattern=6]", shows the BDiagonal pattern on the object's background.



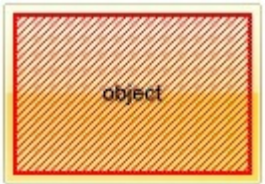
- "[frame=RGB(255,0,0),framethick]", draws a red thick-border around the object.



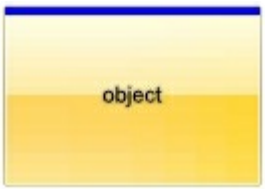
- "[frame=RGB(255,0,0),framethick,pattern=6,patterncolor=RGB(255,0,0)]", draws a red thick-border around the object, with a patter inside.



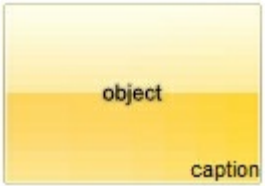
- "[[patterncolor=RGB(255,0,0)] (none[(4,4,100%-8,100%-8),pattern=0x006,patterncolor=RGB(255,0,0),frame=RGB(25 draws a red thick-border around the object, with a patter inside, with a 4-pixels wide padding:



- "top[4,back=RGB(0,0,255)]", draws a blue line on the top side of the object's background, of 4-pixels wide.

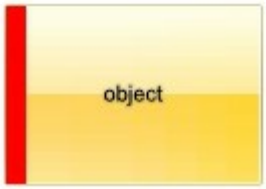


- "[text=`caption`,align=0x22]", shows the caption string aligned to the bottom-right side of the object's background.

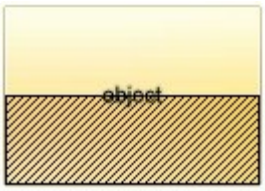


- "[text=`<img>flag</img>`,align=0x11]" shows the flag picture and the sweden string aligned to the bottom side of the object.

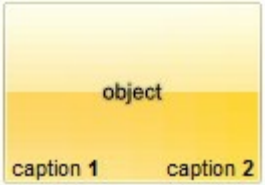
- "left[10,back=RGB(255,0,0)]", draws a red line on the left side of the object's background, of 10-pixels wide.



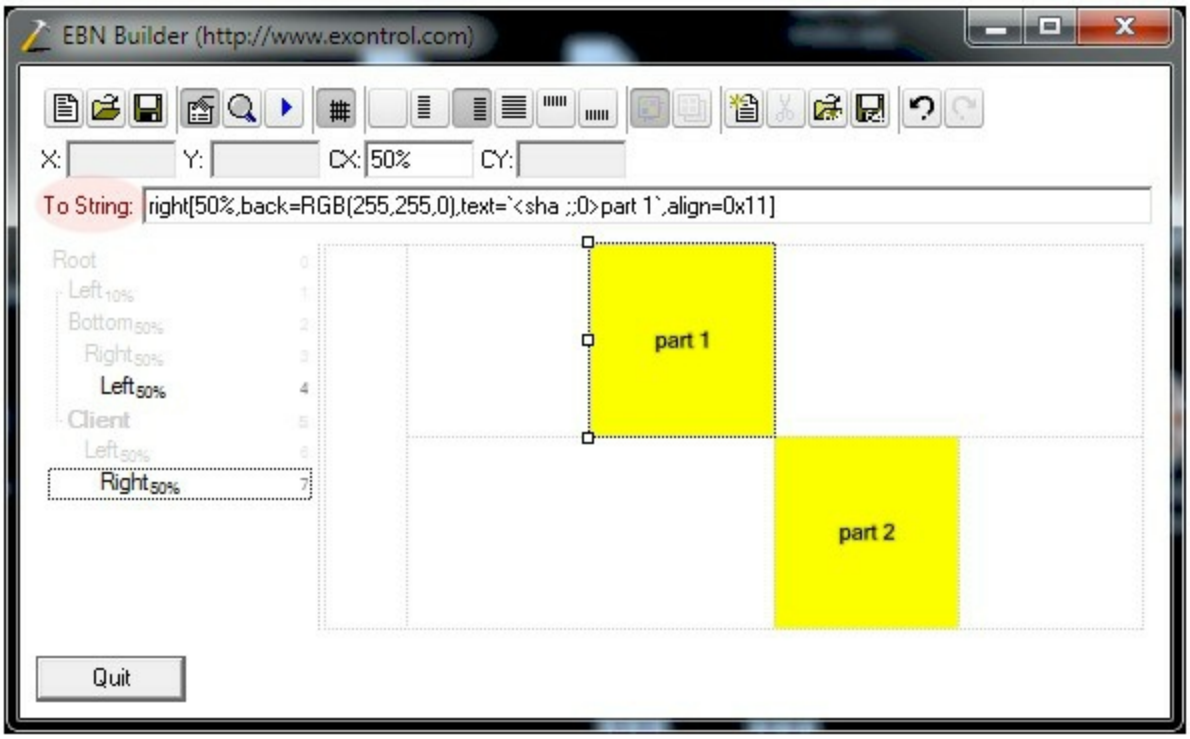
- "bottom[50%,pattern=6,frame]", shows the BDiagonal pattern with a border arround on the lower-half part of the object's background.



- "root[text=`caption <b>2`,align=0x22](client[text=`caption <b>1`,align=0x20])", shows the caption **1** aligned to the bottom-left side, and the caption **2** to the bottom-right side



*The Exontrol's [eXButton](#) WYSWYG Builder helps you to generate or view the EBN String Format, in the **To String** field as shown in the following screen shot:*



The **To String** field of the EBN Builder defines the **EBN String Format** that can be used on BackgroundExt property.

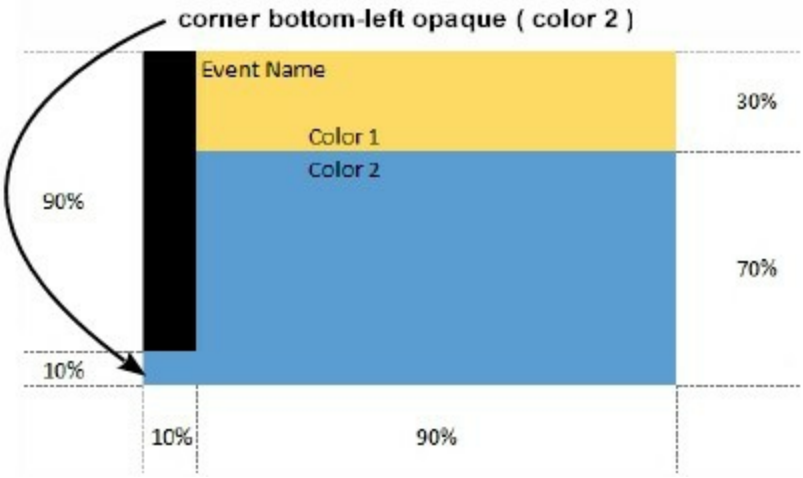
The **EBN String Format** syntax in BNF notation is defined like follows:

```
<EBN> ::= <elements> | <root> "(" [<elements>] ")"
<elements> ::= <element> [ "," <elements> ]
<root> ::= "root" [ <attributes> ] | [ <attributes> ]
<element> ::= <anchor> [ <attributes> ] [ "(" [<elements>] ")" ]
<anchor> ::= "none" | "left" | "right" | "client" | "top" | "bottom"
<attributes> ::= "[" [<client> ","] <attribute> [ "," <attributes> ] "]"
<client> ::= <expression> | <expression> "," <expression> "," <expression> ","
<expression>
<expression> ::= <number> | <number> "%"
<attribute> ::= <backcolor> | <text> | <wordwrap> | <align> | <pattern> |
<patterncolor> | <frame> | <framethick> | <data> | <others>
<equal> ::= "="
```

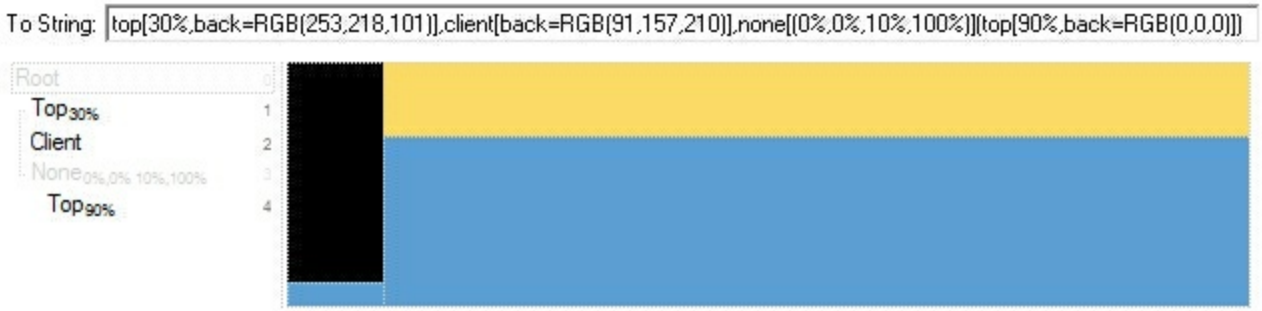
```
<digit> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
<decimal> ::= <digit><decimal>
<hexadigit> ::= <digit> | "A" | "B"  "C" | "D" | "E"  "F"
<hexa> ::= <hexadigit><hexa>
<number> ::= <decimal> | "0x" <hexa>
<color> ::= <rgbcolor> | number
<rgbcolor> ::= "RGB" "(" <number> "," <number> "," <number> ")"
<string> ::= "`" <characters> "`" | "'" <characters> "'" | " <characters> "
<characters> ::= <char>|<characters>
<char> ::= <any_character_excepts_null>
<backcolor> ::= "back" <equal> <color>
<text> ::= "text" <equal> <string>
<align> ::= "align" <equal> <number>
<pattern> ::= "pattern" <equal> <number>
<patterncolor> ::= "patterncolor" <equal> <color>
<frame> ::= "frame" <equal> <color>
<data> ::= "data" <equal> <number> | <string>
<framethick> ::= "framethick"
<wordwrap> ::= "wordwrap"
```

Others like: pic, stretch, hstretch, vstretch, transparent, from, to are reserved for future use only.

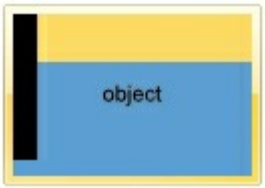
Now, lets say we have the following request to layout the colors on the objects:



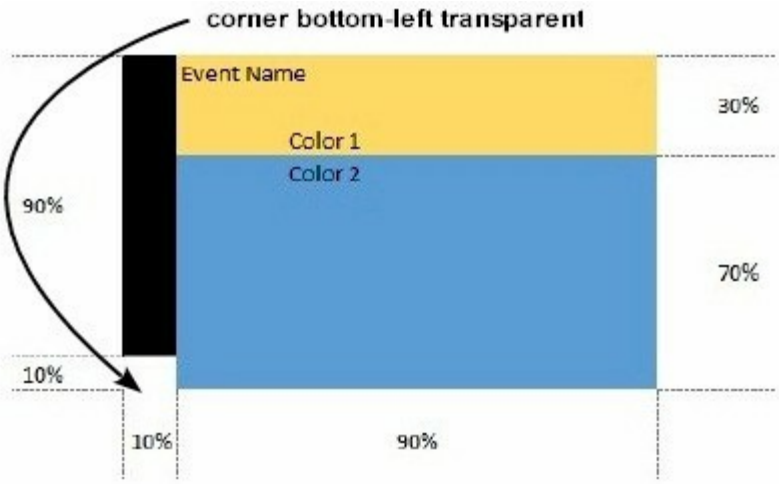
We define the BackgroundExt property such as "top[30%,back=RGB(253,218,101)],client[back=RGB(91,157,210)],none[(0%,0%,10%,100
(top[90%,back=RGB(0,0,0)])", and it looks as:



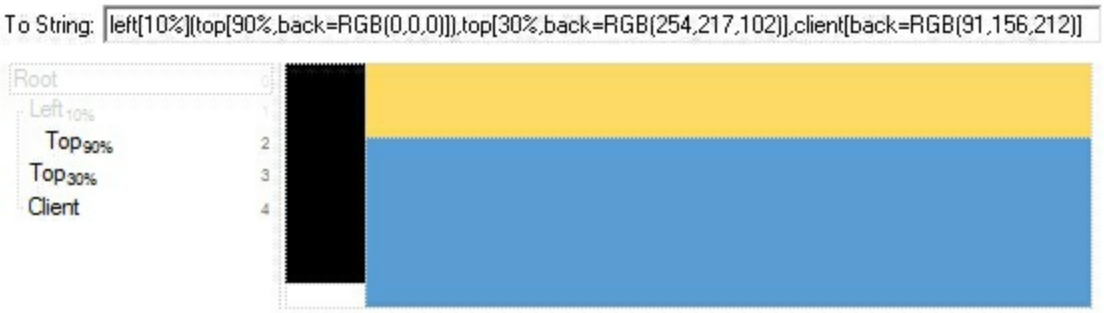
so, if we apply to our object we got:



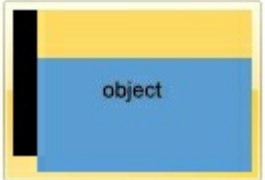
Now, lets say we have the following request to layout the colors on the objects:



We define BackgroundExt property such as "left[10%](top[90%,back=RGB(0,0,0)]),top[30%,back=RGB(254,217,102)],client[back=RGB(91,156,2
and it looks as:

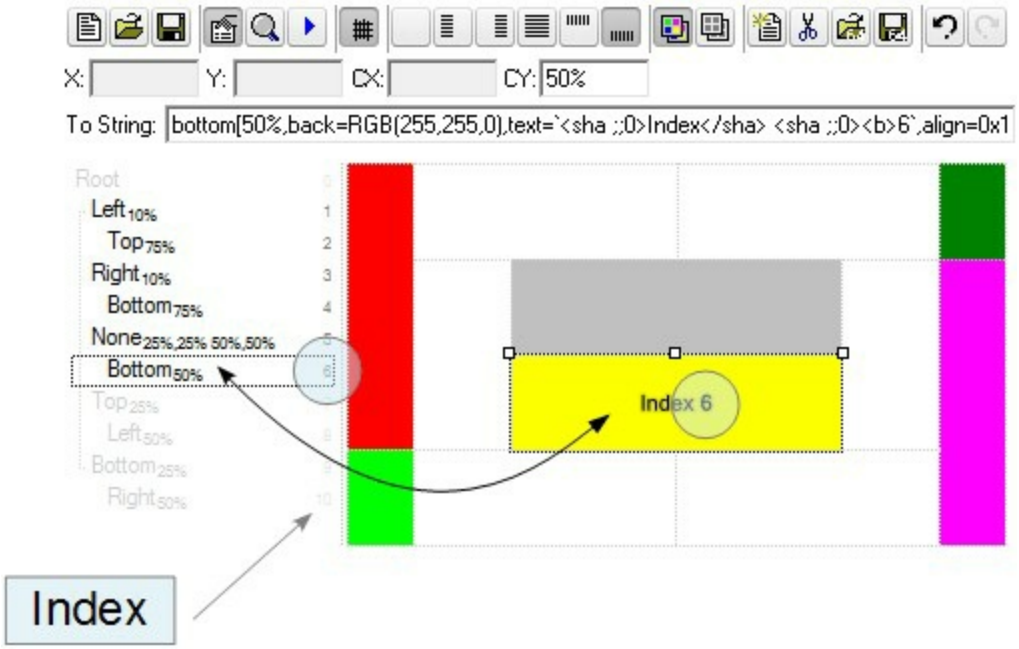


so, if we apply to our object we got:


object

property Panel.BackgroundExtValue(Index as IndexExtEnum, Property as BackgroundExtPropertyEnum) as Variant

Specifies at runtime, the value of the giving property for specified part of the background extension.

Type	Description
Index as IndexExtEnum	A Long expression that defines the index of the part that composes the EBN to be accessed / changed. The following screen shot shows where you can find Index of the parts:  The screen shot shows that the EBN contains 11 elements, so in this case the Index starts at 0 (root element) and ends on 10.
Property as BackgroundExtPropertyEnum	A BackgroundExtPropertyEnum expression that specifies the property to be changed as explained bellow.
Variant	A Variant expression that defines the part's value. The Type of the expression depending on the Property parameter as explained bellow.

Use the BackgroundExtValue property to change at runtime any UI property for any part that composes the EBN String Format. The BackgroundExtValue property has no effect if the BackgroundExt property is empty (by default). *The idea is as follows: first you need to decide the layout of the UI to put on the object's background, using the BackgroundExt*

property, and next (if required), you can change any property of any part of the background extension to a new value. In other words, let's say you have the same layout to be applied to some of your objects, so you specify the BackgroundExt to be the same for them, and next use the BackgroundExtValue property to change particular properties (like back-color, size, position, anchor) for different objects.

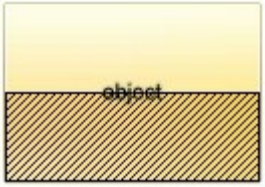
You can access/define/change the following UI properties of the element:

- **exBackColorExt**(1), Indicates the background color / EBN color to be shown on the part of the object. *Sample: 255 indicates red, RGB(0,255,0) green, or 0x1000000. (Color/Numeric expression, The last 7 bits in the high significant byte of the color indicate the identifier of the skin being used)*
- **exClientExt**(2), Specifies the position/size of the object, depending on the object's anchor. The syntax of the exClientExt is related to the exAnchorExt value. *For instance, if the object is anchored to the left side of the parent (exAnchorExt = 1), the exClientExt specifies just the width of the part in pixels/percents, not including the position. In case, the exAnchorExt is client, the exClientExt has no effect. Sample: 50% indicates half of the parent, 25 indicates 25 pixels, or 50%-8 indicates 8-pixels left from the center of the parent. (String/Numeric expression)*
- **exAnchorExt**(3), Specifies the object's alignment relative to its parent. *(Numeric expression)*
- **exTextExt**(4), Specifies the HTML text to be displayed on the object. *(String expression)*
- **exTextExtWordWrap**(5), Specifies that the object is wrapping the text. The exTextExt value specifies the HTML text to be displayed on the part of the EBN object. This property has effect only if there is a text assigned to the part using the exTextExt flag. *(Boolean expression)*
- **exTextExtAlignment**(6), Indicates the alignment of the text on the object. The exTextExt value specifies the HTML text to be displayed on the part of the EBN object. This property has effect only if there is a text assigned to the part using the exTextExt flag *(Numeric expression)*
- **exPatternExt**(7), Indicates the pattern to be shown on the object. The exPatternColorExt specifies the color to show the pattern. *(Numeric expression)*
- **exPatternColorExt**(8), Indicates the color to show the pattern on the object. The exPatternColorExt property has effect only if the exPatternExt property is not 0 (empty). The exFrameColorExt specifies the color to show the frame (the exPatternExt property includes the exFrame or exFrameThick flag). *(Color expression)*
- **exFrameColorExt**(9), Indicates the color to show the border-frame on the object. This property set the Frame flag for exPatternExt property. *(Color expression)*
- **exFrameThickExt**(11), Specifies that a thick-frame is shown around the object. This property set the FrameThick flag for exPatternExt property. *(Boolean expression)*
- **exUserDataExt**(12), Specifies an extra-data associated with the object. *(Variant*

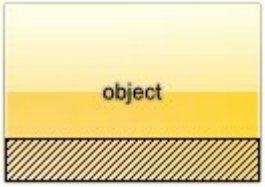
expression)

For instance, having the BackgroundExt on "bottom[50%,pattern=6,frame]"

we got:



so let's change the percent of 50% to 25% like BackgroundExtValue(1,2) on "25%", where 1 indicates the first element after root, and 2 indicates the **exClientExt** property, we get:



In VB you should have the following syntax:

```
.BackgroundExt = "bottom[50%,pattern=6,frame]"
.BackgroundExtValue(exIndexExt1, exClientExt) = "25%"
```

property Panel.Bold as Boolean

Specifies whether the text in the panel appears in bold.

Type	Description
Boolean	A Boolean expression that specifies whether the caption of the panel should be shown in bold.

Use the Bold property to show the panel's caption in bold. Use the Text property to assign a caption or an HTML text to a panel. Use the <b> HTML tag to show in bold only a portion of the caption in the panel. Use the Italic property to show the panel's caption in italic. Use the Underline property to underline the panel's caption. Use the StrikeOut property to show the panel's caption in strikeout.

The following **VB** sample shows "How can I show in bold a specified panel":

```
With StatusBar1
   .BeginUpdate
   .VisualAppearance.Add 4,"c:\exontrol\images\border.ebn"
   .VisualAppearance.Add 5,"CP:4 1 1 -1 -1"
   .BackColorPanels = 83886080
   .BackColor = -2147483633
   .Format = "1/2"
   With .Panel(1)
      .Text = "Panel 1"
      .Bold = True
   End With
   .Panel(2).Text = "<b>Panel</b> 2"
   .EndUpdate
End With
```

The following **VB.NET** sample shows "How can I show in bold a specified panel":

```
With AxStatusBar1
   .BeginUpdate
   .VisualAppearance.Add 4,"c:\exontrol\images\border.ebn"
   .VisualAppearance.Add 5,"CP:4 1 1 -1 -1"
   .GetOcx().BackColorPanels = &H5000000
   .GetOcx().BackColor = &H8000000f
   .Format = "1/2"
```

```
    With .get_Panel(1)
        .Text = "Panel 1"
        .Bold = True
    End With
    .get_Panel(2).Text = "<b>Panel</b> 2"
    .EndUpdate
End With
```

The following **C++** sample shows "How can I show in bold a specified panel":

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

    #import "C:\\WINNT\\system32\\ExStatusBar.dll"
    using namespace EXSTATUSBARLib;
*/
EXSTATUSBARLib::IStatusBarPtr spStatusBar1 = GetDlgItem(IDC_STATUSBAR1)-
>GetControlUnknown();
spStatusBar1->BeginUpdate();
spStatusBar1->GetVisualAppearance()->Add(4,"c:\\exontrol\\images\\border.ebn");
spStatusBar1->GetVisualAppearance()->Add(5,"CP:4 1 1 -1 -1");
spStatusBar1->PutBackColorPanels(83886080);
spStatusBar1->PutBackColor(-2147483633);
spStatusBar1->PutFormat(L"1/2");
EXSTATUSBARLib::IPanelPtr var_Panel = spStatusBar1->GetPanel(long(1));
    var_Panel->PutText(L"Panel 1");
    var_Panel->PutBold(VARIANT_TRUE);
spStatusBar1->GetPanel(long(2))->PutText(L"<b>Panel</b> 2");
spStatusBar1->EndUpdate();
```

The following **C#** sample shows "How can I show in bold a specified panel":

```
axStatusBar1.BeginUpdate();
axStatusBar1.VisualAppearance.Add(4,"c:\\exontrol\\images\\border.ebn");
axStatusBar1.VisualAppearance.Add(5,"CP:4 1 1 -1 -1");
(axStatusBar1.GetOcx() as EXSTATUSBARLib.StatusBar).BackColorPanels = 0x5000000;
```

```
(axStatusBar1.GetOcx() as EXSTATUSBARLib.StatusBar).BackColor = 0x8000000f;
axStatusBar1.Format = "1/2";
EXSTATUSBARLib.Panel var_Panel = axStatusBar1.get_Panel(1);
   var_Panel.Text = "Panel 1";
   var_Panel.Bold = true;
axStatusBar1.get_Panel(2).Text = "<b>Panel</b> 2";
axStatusBar1.EndUpdate();
```

The following **VFP** sample shows "How can I show in bold a specified panel":

```
with thisform.StatusBar1
   .BeginUpdate
   .VisualAppearance.Add(4,"c:\exontrol\images\border.ebn")
   .VisualAppearance.Add(5,"CP:4 1 1 -1 -1")
   .BackColorPanels = 83886080
   .BackColor = -2147483633
   .Format = "1/2"
   with .Panel(1)
      .Text = "Panel 1"
      .Bold = .T.
   endwith
   .Panel(2).Text = "<b>Panel</b> 2"
   .EndUpdate
endwith
```

property Panel.ControlID as String

Specifies the program identifier being shown in the panel.

Type	Description
String	A string expression that can be formatted as follows: a prog ID, a CLSID, a URL, a reference to an Active document , a fragment of HTML.

The control supports ActiveX hosting, so you can insert any ActiveX component. The License property specifies the runtime license required to create the inner ActiveX control. Runtime-less controls are not requiring a runtime key so there is not need to call the License property before calling the ControlID property. Use the Object property to access the properties and methods of the inner ActiveX control. The control fires the OleEvent event to notify when an inner control fires an event.

The ControlID must be formatted in one of the following ways:

- A ProgID such as "Exontrol.Tree"
- A CLSID such as "{8E27C92B-1264-101C-8A2F-040224009C02}"
- A URL such as "https://www.exontrol.com"
- A reference to an Active document such as "c:\temp\myfile.doc", or "c:\temp\picture.gif"
- A fragment of HTML such as "MSHTML:<HTML><BODY>This is a line of text</BODY></HTML>"
- A fragment of XML

The ControlID property creates an ActiveX control that's hosted by the control. **The look and feel of the inner ActiveX control depends on the identifier you are using, and the version of the library that implements the ActiveX control, so you need to consult the documentation of the inner ActiveX control you are inserting inside the control.**

The following **VB** sample shows "How do I insert an ActiveX control to a panel":

```
With StatusBar1
   .BeginUpdate
   .Images
"gBJJgBAIDAAGAAEAAQhYAf8Pf4hh0QihCJo2AEZjQAjEZFEaIEaEEaAIAkcbk0oIUrlktl0vmExn
 & _
"/oFBoVDoIFo1HpFJpVLplNp1PqFRqVTqlVq1XrFZrVbrldr1fsFhsVjslls1ntFptVrtltt1vuFxuVzul
 & _
"/wGBwWDwmFw2HxGJxWLxmNx0xiFdyOTh8Tf9ZymXx+QytcyNgz8r0OblWjyWds+m0ka1
```

```
 & _
"x3GO4NV3WeyvD2XJ5XL5nN51aiw+lfSj0gkUkAEllHanHI5j/cHg8EZf7w8vl8j4f/qfEZeB09/v
 & _
"NAOAEAwCjMBwFAEDwJBMDwLBYAP2/8Hv8/gAGAD8LQs9w/nhDY/oygIA="
   With .VisualAppearance
      .Add 4,"c:\exontrol\images\border.ebn"
      .Add 5,"CP:4 1 1 -1 -1"
   End With
   .BackColorPanels = 83886080
   .BackColor = -2147483633
   .Format = "1/2,(24;5/6)"
   .Panel(1).ControlID = "MSChart20Lib.MSChart"
   .Panel(2).ControlID = "MSCAL.Calendar"
   With .Panel(5)
      .Text = "<fgcolor=FFFFFF><img>1</img>75%</fgcolor>"
      .Alignment = exAlignMiddleLeft
      .Percent = 75
      .Transparency = 35
      .Offset = "4 2 -4 -2"
   End With
   .EndUpdate
End With
```

The following **VB.NET** sample shows "How do I insert an ActiveX control to a panel":

```
With AxStatusBar1
   .BeginUpdate
   .Images
"gBJJgBAIDAAGAAEAAQhYAf8Pf4hh0QihCJo2AEZjQAjEZFEaIEaEEaAIAkcbk0oIUrlktl0vmExn
 & _
"/oFBoVDoIFo1HpFJpVLpINp1PqFRqVTqIVq1XrFZrVbrIdr1fsFhsVjslls1ntFptVrtltt1vuFxuVzul
 & _
"/wGBwWDwmFw2HxGJxWLxmNx0xiFdyOTh8Tf9ZymXx+QytcyNgz8r0OblWjyWds+m0ka1
 & _
"x3GO4NV3WeyvD2XJ5XL5nN51aiw+lfSj0gkUkAEllHanHI5j/cHg8EZf7w8vl8j4f/qfEZeB09/v
 & _
"NAOAEAwCjMBwFAEDwJBMDwLBYAP2/8Hv8/gAGAD8LQs9w/nhDY/oygIA="
```

```
    With .VisualAppearance
        .Add 4,"c:\exontrol\images\border.ebn"
        .Add 5,"CP:4 1 1 -1 -1"
    End With
    .GetOcx().BackColorPanels = &H5000000
    .GetOcx().BackColor = &H8000000f
    .Format = "1/2,(24;5/6)"
    .get_Panel(1).ControlID = "MSChart20Lib.MSChart"
    .get_Panel(2).ControlID = "MSCAL.Calendar"
    With .get_Panel(5)
        .Text = "<fgcolor=FFFFFF><img>1</img>75%</fgcolor>"
        .Alignment = EXSTATUSBARLib.TextAlignEnum.exAlignMiddleLeft
        .Percent = 75
        .Transparency = 35
        .Offset = "4 2 -4 -2"
    End With
    .EndUpdate
End With
```

The following **C++** sample shows "How do I insert an ActiveX control to a panel":

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

    #import "D:\\Exontrol\\ExStatusBar\\project\\Demo\\ExStatusBar.dll"
    using namespace EXSTATUSBARLib;
*/
EXSTATUSBARLib::IStatusBarPtr spStatusBar1 = GetDlgItem(IDC_STATUSBAR1)-
>GetControlUnknown();
spStatusBar1->BeginUpdate();
spStatusBar1-
>Images(_bstr_t("gBJJgBAIDAAGAAEAAQhYAf8Pf4hh0QihCJo2AEZjQAjEZFEaIEaEEaAIAkcbk
 +
"/oFBoVDoIFo1HpFJpVLpINp1PqFRqVTqIVq1XrFZrVbrIdr1fsFhsVjslls1ntFptVrtltt1vuFxuVzul
 +
```

```
"/wGBwWDwmFw2HxGJxWLxmNx0xiFdyOTh8Tf9ZymXx+QytcyNgz8r0OblWjyWds+m0ka1
 +
"x3GO4NV3WeyvD2XJ5XL5nN51aiw+lfSj0gkUkAEllHanHI5j/cHg8EZf7w8vl8j4f/qfEZeB09/v
 +
"NAOAEAwCjMBwFAEDwJBMDwLBYAP2/8Hv8/gAGAD8LQs9w/nhDY/oygIA=");
EXSTATUSBARLib::IAppearancePtr var_Appearance = spStatusBar1->GetVisualAppearance();
   var_Appearance->Add(4,"c:\\exontrol\\images\\border.ebn");
   var_Appearance->Add(5,"CP:4 1 1 -1 -1");
spStatusBar1->PutBackColorPanels(83886080);
spStatusBar1->PutBackColor(-2147483633);
spStatusBar1->PutFormat(L"1/2,(24;5/6)");
spStatusBar1->GetPanel(long(1))->PutControlID(L"MSChart20Lib.MSChart");
spStatusBar1->GetPanel(long(2))->PutControlID(L"MSCAL.Calendar");
EXSTATUSBARLib::IPanelPtr var_Panel = spStatusBar1->GetPanel(long(5));
   var_Panel->PutText(L"<fgcolor=FFFFFF><img>1</img>75%</fgcolor>");
   var_Panel->PutAlignment(EXSTATUSBARLib::exAlignMiddleLeft);
   var_Panel->PutPercent(75);
   var_Panel->PutTransparency(35);
   var_Panel->PutOffset(L"4 2 -4 -2");
spStatusBar1->EndUpdate();
```

The following **C#** sample shows "How do I insert an ActiveX control to a panel":

```
axStatusBar1.BeginUpdate();
axStatusBar1.Images("gBJJgBAIDAAGAAEAAQhYAf8Pf4hh0QihCJo2AEZjQAjEZFEaIEaEEaAIA
 +
"/oFBoVDoIFo1HpFJpVLpINp1PqFRqVTqIVq1XrFZrVbrIdr1fsFhsVjslls1ntFptVrtltt1vuFxuVzul
 +
"/wGBwWDwmFw2HxGJxWLxmNx0xiFdyOTh8Tf9ZymXx+QytcyNgz8r0OblWjyWds+m0ka1
 +
"x3GO4NV3WeyvD2XJ5XL5nN51aiw+lfSj0gkUkAEllHanHI5j/cHg8EZf7w8vl8j4f/qfEZeB09/v
 +
"NAOAEAwCjMBwFAEDwJBMDwLBYAP2/8Hv8/gAGAD8LQs9w/nhDY/oygIA=");
EXSTATUSBARLib.Appearance var_Appearance = axStatusBar1.VisualAppearance;
   var_Appearance.Add(4,"c:\\exontrol\\images\\border.ebn");
   var_Appearance.Add(5,"CP:4 1 1 -1 -1");
```

```
(axStatusBar1.GetOcx() as EXSTATUSBARLib.StatusBar).BackColorPanels = 0x5000000;
(axStatusBar1.GetOcx() as EXSTATUSBARLib.StatusBar).BackColor = 0x8000000f;
axStatusBar1.Format = "1/2,(24;5/6)";
axStatusBar1.get_Panel(1).ControlID = "MSChart20Lib.MSChart";
axStatusBar1.get_Panel(2).ControlID = "MSCAL.Calendar";
EXSTATUSBARLib.Panel var_Panel = axStatusBar1.get_Panel(5);
   var_Panel.Text = "<fgcolor=FFFFFF><img>1</img>75%</fgcolor>";
   var_Panel.Alignment = EXSTATUSBARLib.TextAlignEnum.exAlignMiddleLeft;
   var_Panel.Percent = 75;
   var_Panel.Transparency = 35;
   var_Panel.Offset = "4 2 -4 -2";
axStatusBar1.EndUpdate();
```

The following **VFP** sample shows "How do I insert an ActiveX control to a panel":

```
with thisform.StatusBar1
   .BeginUpdate
   var_s =
"gBJJgBAIDAAGAAEAAQhYAf8Pf4hh0QihCJo2AEZjQAjEZFEaIEaEEaAIAkcbk0oIUrlktl0vmExn

   var_s = var_s +
"oFBoVDoIFo1HpFJpVLplNp1PqFRqVTqlVq1XrFZrVbrldr1fsFhsVjslls1ntFptVrtltt1vuFxuVzul1

   var_s = var_s +
"wGBwWDwmFw2HxGJxWLxmNx0xiFdyOTh8Tf9ZymXx+QytcyNgz8r0OblWjyWds+m0ka1V

   var_s = var_s +
"3GO4NV3WeyvD2XJ5XL5nN51aiw+lfSj0gkUkAEllHanHI5j/cHg8EZf7w8vl8j4f/qfEZeB09/vj

   var_s = var_s +
"AOAEAwCjMBwFAEDwJBMDwLBYAP2/8Hv8/gAGAD8LQs9w/nhDY/oygIA="
   .Images(var_s)
   with .VisualAppearance
      .Add(4,"c:\exontrol\images\border.ebn")
      .Add(5,"CP:4 1 1 -1 -1")
   endwith
   .BackColorPanels = 83886080
```

```
    .BackColor = -2147483633
    .Format = "1/2,(24;5/6)"
    .Panel(1).ControlID = "MSChart20Lib.MSChart"
    .Panel(2).ControlID = "MSCAL.Calendar"
    with .Panel(5)
        .Text = "<fgcolor=FFFFFF><img>1</img>75%</fgcolor>"
        .Alignment = 16
        .Percent = 75
        .Transparency = 35
        .Offset = "4 2 -4 -2"
    endwith
    .EndUpdate
endwith
```

property Panel.Enabled as Boolean

Specifies whether the panel is enabled or disabled.

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

By default, the Enabled property is True. Use the Enabled property to enable or disable a panel. Use the Enabled property to disable the entire control. Use the ForeColor property to specify the panel's foreground color when the panel is enabled.

The following **VB** sample shows "How can I disable a panel":

```
With StatusBar1
   .BeginUpdate
   .VisualAppearance.Add 4,"c:\exontrol\images\border.ebn"
   .VisualAppearance.Add 5,"CP:4 1 1 -1 -1"
   .BackColorPanels = 83886080
   .BackColor = -2147483633
   .Format = "1/2,(5/6)"
   With .Panel(5)
      .Text = "Disabled"
      .Enabled = False
   End With
   .EndUpdate
End With
```

The following **VB.NET** sample shows "How can I disable a panel":

```
With AxStatusBar1
   .BeginUpdate
   .VisualAppearance.Add 4,"c:\exontrol\images\border.ebn"
   .VisualAppearance.Add 5,"CP:4 1 1 -1 -1"
   .GetOcx().BackColorPanels = &H5000000
   .GetOcx().BackColor = &H8000000f
   .Format = "1/2,(5/6)"
   With .get_Panel(5)
      .Text = "Disabled"
      .Enabled = False
```

```
    End With
    .EndUpdate
End With
```

The following **C++** sample shows "How can I disable a panel":

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

    #import "C:\\WINNT\\system32\\ExStatusBar.dll"
    using namespace EXSTATUSBARLib;
*/
EXSTATUSBARLib::IStatusBarPtr spStatusBar1 = GetDlgItem(IDC_STATUSBAR1)-
>GetControlUnknown();
spStatusBar1->BeginUpdate();
spStatusBar1->GetVisualAppearance()->Add(4,"c:\\exontrol\\images\\border.ebn");
spStatusBar1->GetVisualAppearance()->Add(5,"CP:4 1 1 -1 -1");
spStatusBar1->PutBackColorPanels(83886080);
spStatusBar1->PutBackColor(-2147483633);
spStatusBar1->PutFormat(L"1/2,(5/6)");
EXSTATUSBARLib::IPanelPtr var_Panel = spStatusBar1->GetPanel(long(5));
    var_Panel->PutText(L"Disabled");
    var_Panel->PutEnabled(VARIANT_FALSE);
spStatusBar1->EndUpdate();
```

The following **C#** sample shows "How can I disable a panel":

```
axStatusBar1.BeginUpdate();
axStatusBar1.VisualAppearance.Add(4,"c:\\exontrol\\images\\border.ebn");
axStatusBar1.VisualAppearance.Add(5,"CP:4 1 1 -1 -1");
(axStatusBar1.GetOcx() as EXSTATUSBARLib.StatusBar).BackColorPanels = 0x5000000;
(axStatusBar1.GetOcx() as EXSTATUSBARLib.StatusBar).BackColor = 0x8000000f;
axStatusBar1.Format = "1/2,(5/6)";
EXSTATUSBARLib.Panel var_Panel = axStatusBar1.get_Panel(5);
    var_Panel.Text = "Disabled";
    var_Panel.Enabled = false;
```

```
axStatusBar1.EndUpdate();
```

The following **VFP** sample shows "How can I disable a panel":

```
with thisform.StatusBar1
   .BeginUpdate
   .VisualAppearance.Add(4,"c:\exontrol\images\border.ebn")
   .VisualAppearance.Add(5,"CP:4 1 1 -1 -1")
   .BackColorPanels = 83886080
   .BackColor = -2147483633
   .Format = "1/2,(5/6)"
   with .Panel(5)
      .Text = "Disabled"
      .Enabled = .F.
   endwith
   .EndUpdate
endwith
```

property Panel.ForeColor as Color

Specifies the foreground color of the text in the panel.

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

Use the ForeColor property to specify the foreground color for a specified panel. Use the ForeColor property to specify the foreground color for the entire control. Use the <fgcolor> property to specify a foreground color for a portion of the caption in the panel. Use the Text property to assign a caption to a panel. Use the BackColor property to assign a background color or a visual appearance to a panel.

The following **VB** sample shows "How can I change the caption's foreground color":

```
With StatusBar1
   .BeginUpdate
   .VisualAppearance.Add 4,"c:\exontrol\images\border.ebn"
   .VisualAppearance.Add 5,"CP:4 1 1 -1 -1"
   .BackColorPanels = 83886080
   .BackColor = -2147483633
   .Format = "1:48/2"
   With .Panel(1)
      .Text = "Panel 1"
      .ForeColor = 65535
   End With
   Set var_Panel = .Panel(2)
   With var_Panel
      .Text = "Panel 2"
      .ForeColor = 16711935
   End With
   .EndUpdate
End With
```

The following **VB.NET** sample shows "How can I change the caption's foreground color":

```
Dim var_Panel
With AxStatusBar1
   .BeginUpdate
```

```
    .VisualAppearance.Add 4,"c:\exontrol\images\border.ebn"
    .VisualAppearance.Add 5,"CP:4 1 1 -1 -1"
    .GetOcx().BackColorPanels = &H5000000
    .GetOcx().BackColor = &H8000000f
    .Format = "1:48/2"
    With .get_Panel(1)
        .Text = "Panel 1"
        .ForeColor = 65535
    End With
    var_Panel = .get_Panel(2)
    With var_Panel
        .Text = "Panel 2"
        .ForeColor = 16711935
    End With
    .EndUpdate
End With
```

The following **C++** sample shows "How can I change the caption's foreground color":

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

    #import "C:\\WINNT\\system32\\ExStatusBar.dll"
    using namespace EXSTATUSBARLib;
*/
EXSTATUSBARLib::IStatusBarPtr spStatusBar1 = GetDlgItem(IDC_STATUSBAR1)-
>GetControlUnknown();
spStatusBar1->BeginUpdate();
spStatusBar1->GetVisualAppearance()->Add(4,"c:\\exontrol\\images\\border.ebn");
spStatusBar1->GetVisualAppearance()->Add(5,"CP:4 1 1 -1 -1");
spStatusBar1->PutBackColorPanels(83886080);
spStatusBar1->PutBackColor(-2147483633);
spStatusBar1->PutFormat(L"1:48/2");
EXSTATUSBARLib::IPanelPtr var_Panel = spStatusBar1->GetPanel(long(1));
    var_Panel->PutText(L"Panel 1");
```

```
   var_Panel->PutForeColor(65535);
EXSTATUSBARLib::IPanelPtr var_Panel = spStatusBar1->GetPanel(long(2));
   var_Panel->PutText(L"Panel 2");
   var_Panel->PutForeColor(16711935);
spStatusBar1->EndUpdate();
```

The following **C#** sample shows "How can I change the caption's foreground color":

```
axStatusBar1.BeginUpdate();
axStatusBar1.VisualAppearance.Add(4,"c:\\exontrol\\images\\border.ebn");
axStatusBar1.VisualAppearance.Add(5,"CP:4 1 1 -1 -1");
(axStatusBar1.GetOcx() as EXSTATUSBARLib.StatusBar).BackColorPanels = 0x5000000;
(axStatusBar1.GetOcx() as EXSTATUSBARLib.StatusBar).BackColor = 0x8000000f;
axStatusBar1.Format = "1:48/2";
EXSTATUSBARLib.Panel var_Panel = axStatusBar1.get_Panel(1);
   var_Panel.Text = "Panel 1";
   var_Panel.ForeColor = 65535;
EXSTATUSBARLib.Panel var_Panel = axStatusBar1.get_Panel(2);
   var_Panel.Text = "Panel 2";
   var_Panel.ForeColor = 16711935;
axStatusBar1.EndUpdate();
```

The following **VFP** sample shows "How can I change the caption's foreground color":

```
with thisform.StatusBar1
   .BeginUpdate
   .VisualAppearance.Add(4,"c:\exontrol\images\border.ebn")
   .VisualAppearance.Add(5,"CP:4 1 1 -1 -1")
   .BackColorPanels = 83886080
   .BackColor = -2147483633
   .Format = "1:48/2"
   with .Panel(1)
      .Text = "Panel 1"
      .ForeColor = 65535
   endwith
   var_Panel = .Panel(2)
   with var_Panel
      .Text = "Panel 2"
```

```
        .ForeColor = 16711935
    endwith
    .EndUpdate
endwith
```

property Panel.Height as Long

Specifies the height in pixels of the panel.

Type	Description
Long	A long expression that specifies the height of the panel, in pixels.

Use the Height property to get the height in pixels of the panel. Use the Width property to specify the width of the panel. Use the Format property to control the width and the height of the panels, using CRD strings.

The following **VB** sample shows "Is there any property to get the width/height of the panel":

```
With StatusBar1
   .BeginUpdate
   .VisualAppearance.Add 4,"c:\exontrol\images\border.ebn"
   .VisualAppearance.Add 5,"CP:4 1 1 -1 -1"
   .BackColorPanels = 83886080
   .BackColor = -2147483633
   .Format = "1/2,(5/6)"
   With .Panel(5)
      .Text = .Width
   End With
   With .Panel(6)
      .Text = .Height
   End With
   .EndUpdate
End With
```

The following **VB.NET** sample shows "Is there any property to get the width/height of the panel":

```
With AxStatusBar1
   .BeginUpdate
   .VisualAppearance.Add 4,"c:\exontrol\images\border.ebn"
   .VisualAppearance.Add 5,"CP:4 1 1 -1 -1"
   .GetOcx().BackColorPanels = &H5000000
   .GetOcx().BackColor = &H8000000f
   .Format = "1/2,(5/6)"
```

```
    With .get_Panel(5)
        .Text = .Width
    End With
    With .get_Panel(6)
        .Text = .Height
    End With
    .EndUpdate
End With
```

The following **C++** sample shows "Is there any property to get the width/height of the panel":

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

    #import "C:\\WINNT\\system32\\ExStatusBar.dll"
    using namespace EXSTATUSBARLib;
*/
EXSTATUSBARLib::IStatusBarPtr spStatusBar1 = GetDlgItem(IDC_STATUSBAR1)-
>GetControlUnknown();
spStatusBar1->BeginUpdate();
spStatusBar1->GetVisualAppearance()->Add(4,"c:\\exontrol\\images\\border.ebn");
spStatusBar1->GetVisualAppearance()->Add(5,"CP:4 1 1 -1 -1");
spStatusBar1->PutBackColorPanels(83886080);
spStatusBar1->PutBackColor(-2147483633);
spStatusBar1->PutFormat(L"1/2,(5/6)");
EXSTATUSBARLib::IPanelPtr var_Panel = spStatusBar1->GetPanel(long(5));
    var_Panel->PutText(_bstr_t(var_Panel->GetWidth()));
EXSTATUSBARLib::IPanelPtr var_Panel1 = spStatusBar1->GetPanel(long(6));
    var_Panel1->PutText(_bstr_t(var_Panel1->GetHeight()));
spStatusBar1->EndUpdate();
```

The following **C#** sample shows "Is there any property to get the width/height of the panel":

```
axStatusBar1.BeginUpdate();
axStatusBar1.VisualAppearance.Add(4,"c:\\exontrol\\images\\border.ebn");
```

```
axStatusBar1.VisualAppearance.Add(5,"CP:4 1 1 -1 -1");
(axStatusBar1.GetOcx() as EXSTATUSBARLib.StatusBar).BackColorPanels = 0x5000000;
(axStatusBar1.GetOcx() as EXSTATUSBARLib.StatusBar).BackColor = 0x8000000f;
axStatusBar1.Format = "1/2,(5/6)";
EXSTATUSBARLib.Panel var_Panel = axStatusBar1.get_Panel(5);
   var_Panel.Text = var_Panel.Width.ToString();
EXSTATUSBARLib.Panel var_Panel1 = axStatusBar1.get_Panel(6);
   var_Panel1.Text = var_Panel1.Height.ToString();
axStatusBar1.EndUpdate();
```

The following **VFP** sample shows "Is there any property to get the width/height of the panel":

```
with thisform.StatusBar1
   .BeginUpdate
   .VisualAppearance.Add(4,"c:\exontrol\images\border.ebn")
   .VisualAppearance.Add(5,"CP:4 1 1 -1 -1")
   .BackColorPanels = 83886080
   .BackColor = -2147483633
   .Format = "1/2,(5/6)"
   with .Panel(5)
      .Text = .Width
   endwith
   with .Panel(6)
      .Text = .Height
   endwith
   .EndUpdate
endwith
```

property Panel.Image as Long

Gets or sets the index of the icon to display within the status bar panel.

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

Use the Image property to assign a single icon to a panel. Use the <img> HTML tag in the Text property to assign multiple icons in the panel. Use the Images method to assign a list of icons being used by control. Use the Enabled property to disable a panel. When the panel is disabled the icons are shown as grayed.

The following **VB** sample shows "How can I insert an icon to a panel":

```
With StatusBar1
   .BeginUpdate
   .Images
"gBJJgBAIDAAGAAEAAQhYAf8Pf4hh0QihCJo2AEZjQAjEZFEaIEaEEaAIAkcbk0olUrlktl0vmExn
 & _
"/oFBoVDolFo1HpFJpVLplNp1PqFRqVTqlVq1XrFZrVbrldr1fsFhsVjslls1ntFptVrtltt1vuFxuVzul
 & _
"/wGBwWDwmFw2HxGJxWLxmNx0xiFdyOTh8Tf9ZymXx+QytcyNgz8r0OblWjyWds+m0ka1
 & _
"x3GO4NV3WeyvD2XJ5XL5nN51aiw+lfSj0gkUkAEllHanHI5j/cHg8EZf7w8vl8j4f/qfEZeB09/v
 & _
"NAOAEAwCjMBwFAEDwJBMDwLBYAP2/8Hv8/gAGAD8LQs9w/nhDY/oygIA="
   .VisualAppearance.Add 4,"c:\exontrol\images\border.ebn"
   .VisualAppearance.Add 5,"CP:4 1 1 -1 -1"
   .BackColorPanels = 83886080
   .Format = "1,2,3,4,(5/6/7/8)"
   .Panel(1).Image = 1
   .EndUpdate
End With
```

The following **VB.NET** sample shows "How can I insert an icon to a panel":

```
With AxStatusBar1
   .BeginUpdate
   .Images
```

```
"gBJJgBAIDAAGAAEAAQhYAf8Pf4hh0QihCJo2AEZjQAjEZFEaIEaEEaAIAkcbk0oIUrlktl0vmExn
 & _
"/oFBoVDoIFo1HpFJpVLpINp1PqFRqVTqlVq1XrFZrVbrldr1fsFhsVjslls1ntFptVrtltt1vuFxuVzul
 & _
"/wGBwWDwmFw2HxGJxWLxmNx0xiFdyOTh8Tf9ZymXx+QytcyNgz8r0OblWjyWds+m0ka1
 & _
"x3GO4NV3WeyvD2XJ5XL5nN51aiw+IfSj0gkUkAEllHanHI5j/cHg8EZf7w8vl8j4f/qfEZeB09/v
 & _
"NAOAEAwCjMBwFAEDwJBMDwLBYAP2/8Hv8/gAGAD8LQs9w/nhDY/oygIA="
   .VisualAppearance.Add 4,"c:\exontrol\images\border.ebn"
   .VisualAppearance.Add 5,"CP:4 1 1 -1 -1"
   .GetOcx().BackColorPanels = &H5000000
   .Format = "1,2,3,4,(5/6/7/8)"
   .get_Panel(1).Image = 1
   .EndUpdate
End With
```

The following **C++** sample shows "How can I insert an icon to a panel":

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

   #import "C:\\WINNT\\system32\\ExStatusBar.dll"
   using namespace EXSTATUSBARLib;
*/
EXSTATUSBARLib::IStatusBarPtr spStatusBar1 = GetDlgItem(IDC_STATUSBAR1)-
>GetControlUnknown();
spStatusBar1->BeginUpdate();
spStatusBar1-
>Images(_bstr_t("gBJJgBAIDAAGAAEAAQhYAf8Pf4hh0QihCJo2AEZjQAjEZFEaIEaEEaAIAkcbk
 +
"/oFBoVDoIFo1HpFJpVLpINp1PqFRqVTqlVq1XrFZrVbrldr1fsFhsVjslls1ntFptVrtltt1vuFxuVzul
 +
"/wGBwWDwmFw2HxGJxWLxmNx0xiFdyOTh8Tf9ZymXx+QytcyNgz8r0OblWjyWds+m0ka1
 +
```

```
"x3GO4NV3WeyvD2XJ5XL5nN51aiw+lfSj0gkUkAEllHanHI5j/cHg8EZf7w8vl8j4f/qfEZeB09/v
 +
"NAOAEAwCjMBwFAEDwJBMDwLBYAP2/8Hv8/gAGAD8LQs9w/nhDY/oygIA=");
spStatusBar1->GetVisualAppearance()->Add(4,"c:\\exontrol\\images\\border.ebn");
spStatusBar1->GetVisualAppearance()->Add(5,"CP:4 1 1 -1 -1");
spStatusBar1->PutBackColorPanels(83886080);
spStatusBar1->PutFormat(L"1,2,3,4,(5/6/7/8)");
spStatusBar1->GetPanel(long(1))->PutImage(1);
spStatusBar1->EndUpdate();
```

The following **C#** sample shows "How can I insert an icon to a panel":

```
axStatusBar1.BeginUpdate();
axStatusBar1.Images("gBJJgBAIDAAGAAEAAQhYAf8Pf4hh0QihCJo2AEZjQAjEZFEaIEaEEaAIA
 +
"/oFBoVDoIFo1HpFJpVLpINp1PqFRqVTqIVq1XrFZrVbrIdr1fsFhsVjslls1ntFptVrtltt1vuFxuVzul
 +
"/wGBwWDwmFw2HxGJxWLxmNx0xiFdyOTh8Tf9ZymXx+QytcyNgz8r0OblWjyWds+m0ka1
 +
"x3GO4NV3WeyvD2XJ5XL5nN51aiw+lfSj0gkUkAEllHanHI5j/cHg8EZf7w8vl8j4f/qfEZeB09/v
 +
"NAOAEAwCjMBwFAEDwJBMDwLBYAP2/8Hv8/gAGAD8LQs9w/nhDY/oygIA=");
axStatusBar1.VisualAppearance.Add(4,"c:\\exontrol\\images\\border.ebn");
axStatusBar1.VisualAppearance.Add(5,"CP:4 1 1 -1 -1");
(axStatusBar1.GetOcx() as EXSTATUSBARLib.StatusBar).BackColorPanels = 0x5000000;
axStatusBar1.Format = "1,2,3,4,(5/6/7/8)";
axStatusBar1.get_Panel(1).Image = 1;
axStatusBar1.EndUpdate();
```

The following **VFP** sample shows "How can I insert an icon to a panel":

```
with thisform.StatusBar1
   .BeginUpdate
   var_s =
"gBJJgBAIDAAGAAEAAQhYAf8Pf4hh0QihCJo2AEZjQAjEZFEaIEaEEaAIAkcbk0oIUrlktl0vmExn

   var_s = var_s +
"oFBoVDoIFo1HpFJpVLpINp1PqFRqVTqIVq1XrFZrVbrIdr1fsFhsVjslls1ntFptVrtltt1vuFxuVzul1
```

```
   var_s = var_s +
"wGBwWDwmFw2HxGJxWLxmNx0xiFdyOTh8Tf9ZymXx+QytcyNgz8r0OblWjyWds+m0ka1V

   var_s = var_s +
"3GO4NV3WeyvD2XJ5XL5nN51aiw+lfSj0gkUkAEllHanHI5j/cHg8EZf7w8vl8j4f/qfEZeB09/vj

   var_s = var_s +
"AOAEAwCjMBwFAEDwJBMDwLBYAP2/8Hv8/gAGAD8LQs9w/nhDY/oygIA="
   .Images(var_s)
   .VisualAppearance.Add(4,"c:\exontrol\images\border.ebn")
   .VisualAppearance.Add(5,"CP:4 1 1 -1 -1")
   .BackColorPanels = 83886080
   .Format = "1,2,3,4,(5/6/7/8)"
   .Panel(1).Image = 1
   .EndUpdate
endwith
```

property Panel.Index as Long

Retrieves the identifier of the panel in the status bar.

Type	Description
Long	A long expression that specifies the index of the panel.

Use the Index property to identify a panel in the status bar. Use the Format property to assign and layout your status bar, using CRD strings. Use the Debug property to display the identifiers of the panels. Use the Text property to assign a caption to a panel.

The following **VB** sample shows "How can I get the index of the panel":

```
With StatusBar1
   .BeginUpdate
   .Appearance = None2
   .VisualAppearance.Add 4,"c:\exontrol\images\border.ebn"
   .VisualAppearance.Add 5,"CP:4 1 1 -1 -1"
   .BackColorPanels = 83886080
   .BackColor = -2147483633
   .Format = "1,2,3,4,(5/6/7/8)"
   .Panel(1).Text = 0.Index
   .EndUpdate
End With
```

The following **VB.NET** sample shows "How can I get the index of the panel":

```
With AxStatusBar1
   .BeginUpdate
   .Appearance = EXSTATUSBARLib.AppearanceEnum.None2
   .VisualAppearance.Add 4,"c:\exontrol\images\border.ebn"
   .VisualAppearance.Add 5,"CP:4 1 1 -1 -1"
   .GetOcx().BackColorPanels = &H5000000
   .GetOcx().BackColor = &H8000000f
   .Format = "1,2,3,4,(5/6/7/8)"
   .get_Panel(1).Text = 0.Index
   .EndUpdate
End With
```

The following **C++** sample shows "How can I get the index of the panel":

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

   #import "C:\\WINNT\\system32\\ExStatusBar.dll"
   using namespace EXSTATUSBARLib;
*/
EXSTATUSBARLib::IStatusBarPtr spStatusBar1 = GetDlgItem(IDC_STATUSBAR1)-
>GetControlUnknown();
spStatusBar1->BeginUpdate();
spStatusBar1->PutAppearance(EXSTATUSBARLib::None2);
spStatusBar1->GetVisualAppearance()->Add(4,"c:\\exontrol\\images\\border.ebn");
spStatusBar1->GetVisualAppearance()->Add(5,"CP:4 1 1 -1 -1");
spStatusBar1->PutBackColorPanels(83886080);
spStatusBar1->PutBackColor(-2147483633);
spStatusBar1->PutFormat(L"1,2,3,4,(5/6/7/8)");
spStatusBar1->GetPanel(long(1))->PutText(_bstr_t(0->GetIndex()));
spStatusBar1->EndUpdate();
```

The following **C#** sample shows "How can I get the index of the panel":

```
axStatusBar1.BeginUpdate();
axStatusBar1.Appearance = EXSTATUSBARLib.AppearanceEnum.None2;
axStatusBar1.VisualAppearance.Add(4,"c:\\exontrol\\images\\border.ebn");
axStatusBar1.VisualAppearance.Add(5,"CP:4 1 1 -1 -1");
(axStatusBar1.GetOcx() as EXSTATUSBARLib.StatusBar).BackColorPanels = 0x5000000;
(axStatusBar1.GetOcx() as EXSTATUSBARLib.StatusBar).BackColor = 0x8000000f;
axStatusBar1.Format = "1,2,3,4,(5/6/7/8)";
axStatusBar1.get_Panel(1).Text = 0.Index.ToString();
axStatusBar1.EndUpdate();
```

The following **VFP** sample shows "How can I get the index of the panel":

```
with thisform.StatusBar1
   .BeginUpdate
   .Appearance = 0
   .VisualAppearance.Add(4,"c:\exontrol\images\border.ebn")
```

```
    .VisualAppearance.Add(5,"CP:4 1 1 -1 -1")
    .BackColorPanels = 83886080
    .BackColor = -2147483633
    .Format = "1,2,3,4,(5/6/7/8)"
    .Panel(1).Text = 0.Index
    .EndUpdate
endwith
```

property Panel.Italic as Boolean

Specifies whether the text in the panel appears in italic.

Type	Description
Boolean	A Boolean expression that specifies whether the caption of the panel should be shown in italic.

Use the Italic property to show the panel's caption in italic. Use the Text property to assign a caption or an HTML text to a panel. Use the <i> HTML tag to show in italic only a portion of the caption in the panel. Use the Bold property to show the panel's caption in bold. Use the Underline property to underline the panel's caption. Use the StrikeOut property to show the panel's caption in strikeout.

The following **VB** sample shows "How can I show in italic a specified panel":

```
With StatusBar1
   .BeginUpdate
   .VisualAppearance.Add 4,"c:\exontrol\images\border.ebn"
   .VisualAppearance.Add 5,"CP:4 1 1 -1 -1"
   .BackColorPanels = 83886080
   .BackColor = -2147483633
   .Format = "1/2"
   With .Panel(1)
      .Text = "Panel 1"
      .Italic = True
   End With
   .Panel(2).Text = "<i>Panel</i> 2"
   .EndUpdate
End With
```

The following **VB.NET** sample shows "How can I show in italic a specified panel":

```
With AxStatusBar1
   .BeginUpdate
   .VisualAppearance.Add 4,"c:\exontrol\images\border.ebn"
   .VisualAppearance.Add 5,"CP:4 1 1 -1 -1"
   .GetOcx().BackColorPanels = &H5000000
   .GetOcx().BackColor = &H8000000f
   .Format = "1/2"
```

```
    With .get_Panel(1)
        .Text = "Panel 1"
        .Italic = True
    End With
    .get_Panel(2).Text = "<i>Panel</i> 2"
    .EndUpdate
End With
```

The following **C++** sample shows "How can I show in italic a specified panel":

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

    #import "C:\\WINNT\\system32\\ExStatusBar.dll"
    using namespace EXSTATUSBARLib;
*/
EXSTATUSBARLib::IStatusBarPtr spStatusBar1 = GetDlgItem(IDC_STATUSBAR1)-
>GetControlUnknown();
spStatusBar1->BeginUpdate();
spStatusBar1->GetVisualAppearance()->Add(4,"c:\\exontrol\\images\\border.ebn");
spStatusBar1->GetVisualAppearance()->Add(5,"CP:4 1 1 -1 -1");
spStatusBar1->PutBackColorPanels(83886080);
spStatusBar1->PutBackColor(-2147483633);
spStatusBar1->PutFormat(L"1/2");
EXSTATUSBARLib::IPanelPtr var_Panel = spStatusBar1->GetPanel(long(1));
    var_Panel->PutText(L"Panel 1");
    var_Panel->PutItalic(VARIANT_TRUE);
spStatusBar1->GetPanel(long(2))->PutText(L"<i>Panel</i> 2");
spStatusBar1->EndUpdate();
```

The following **C#** sample shows "How can I show in italic a specified panel":

```
axStatusBar1.BeginUpdate();
axStatusBar1.VisualAppearance.Add(4,"c:\\exontrol\\images\\border.ebn");
axStatusBar1.VisualAppearance.Add(5,"CP:4 1 1 -1 -1");
(axStatusBar1.GetOcx() as EXSTATUSBARLib.StatusBar).BackColorPanels = 0x5000000;
```

```
(axStatusBar1.GetOcx() as EXSTATUSBARLib.StatusBar).BackColor = 0x8000000f;
axStatusBar1.Format = "1/2";
EXSTATUSBARLib.Panel var_Panel = axStatusBar1.get_Panel(1);
   var_Panel.Text = "Panel 1";
   var_Panel.Italic = true;
axStatusBar1.get_Panel(2).Text = "<i>Panel</i> 2";
axStatusBar1.EndUpdate();
```

The following **VFP** sample shows "How can I show in italic a specified panel":

```
with thisform.StatusBar1
   .BeginUpdate
   .VisualAppearance.Add(4,"c:\exontrol\images\border.ebn")
   .VisualAppearance.Add(5,"CP:4 1 1 -1 -1")
   .BackColorPanels = 83886080
   .BackColor = -2147483633
   .Format = "1/2"
   with .Panel(1)
      .Text = "Panel 1"
      .Italic = .T.
   endwith
   .Panel(2).Text = "<i>Panel</i> 2"
   .EndUpdate
endwith
```

property Panel.License as String

Specifies the runtime license required to create the user control inside the panel.

Type	Description
String	A String expression that specifies the runtime key required to create an inner ActiveX control.

The control supports ActiveX hosting, so you can insert any ActiveX component. Use the ControlID property to specify the program identifier being shown in the panel. Runtime-less controls are not requiring a runtime key so there is not need to call the License property before calling the ControlID property. Use the Object property to access the properties and methods of the inner ActiveX control. The control fires the OleEvent event to notify when an inner control fires an event. The ControlID property creates an ActiveX control that's hosted by the control. **The look and feel of the inner ActiveX control depends on the identifier you are using, and the version of the library that implements the ActiveX control, so you need to consult the documentation of the inner ActiveX control you are inserting inside the control.**

property Panel.Object as Object

Retrieves the inside control being created by ControlID property.

Type	Description
Object	An Object that indicates the inner ActiveX control being created by ControlID property.

Use the Object property to access the properties and methods of the inner ActiveX control. The control supports ActiveX hosting, so you can insert any ActiveX component. The License property specifies the runtime license required to create the inner ActiveX control. Runtime-less controls are not requiring a runtime key so there is not need to call the License property before calling the ControlID property. The control fires the OleEvent event to notify when an inner control fires an event. The ControlID property creates an ActiveX control that's hosted by the control. **The look and feel of the inner ActiveX control depends on the identifier you are using, and the version of the library that implements the ActiveX control, so you need to consult the documentation of the inner ActiveX control you are inserting inside the control.**

The following **VB** sample shows "How do I access the properties and the methods of an inner ActiveX control to a panel":

```
With StatusBar1
   .BeginUpdate
   .Images
"gBJJgBAIDAAGAAEAAQhYAf8Pf4hh0QihCJo2AEZjQAjEZFEaIEaEEaAIAkcbk0oIUrlktl0vmExn
 & _
"/oFBoVDoIFo1HpFJpVLpINp1PqFRqVTqlVq1XrFZrVbrldr1fsFhsVjslls1ntFptVrtltt1vuFxuVzul
 & _
"/wGBwWDwmFw2HxGJxWLxmNx0xiFdyOTh8Tf9ZymXx+QytcyNgz8r0OblWjyWds+m0ka1
 & _
"x3GO4NV3WeyvD2XJ5XL5nN51aiw+lfSj0gkUkAEllHanHI5j/cHg8EZf7w8vl8j4f/qfEZeB09/v
 & _
"NAOAEAwCjMBwFAEDwJBMDwLBYAP2/8Hv8/gAGAD8LQs9w/nhDY/oygIA="
   With .VisualAppearance
      .Add 4,"c:\exontrol\images\border.ebn"
      .Add 5,"CP:4 1 1 -1 -1"
   End With
   .BackColorPanels = 83886080
   .BackColor = -2147483633
```

```
    .Format = "1/2,(24;5/6)"
    With .Panel(1)
        .ControlID = "MSCAL.Calendar"
        With .Object
            .MonthLength = 0
            .BackColor = 16777215
        End With
    End With
    With .Panel(5)
        .Text = "<fgcolor=FFFFFF><img>1</img>75%</fgcolor>"
        .Alignment = exAlignMiddleLeft
        .Percent = 75
        .Transparency = 35
        .Offset = "4 2 -4 -2"
    End With
    .EndUpdate
End With
```

The following **VB.NET** sample shows "How do I access the properties and the methods of an inner ActiveX control to a panel":

```
With AxStatusBar1
    .BeginUpdate
    .Images
"gBJJgBAIDAAGAAEAAQhYAf8Pf4hh0QihCJo2AEZjQAjEZFEaIEaEEaAIAkcbk0oIUrlktl0vmExn
 & _
"/oFBoVDoIFo1HpFJpVLplNp1PqFRqVTqlVq1XrFZrVbrldr1fsFhsVjslls1ntFptVrtltt1vuFxuVzul
 & _
"/wGBwWDwmFw2HxGJxWLxmNx0xiFdyOTh8Tf9ZymXx+QytcyNgz8r0OblWjyWds+m0ka1
 & _
"x3GO4NV3WeyvD2XJ5XL5nN51aiw+lfSj0gkUkAEllHanHI5j/cHg8EZf7w8vl8j4f/qfEZeB09/v
 & _
"NAOAEAwCjMBwFAEDwJBMDwLBYAP2/8Hv8/gAGAD8LQs9w/nhDY/oygIA="
    With .VisualAppearance
        .Add 4,"c:\exontrol\images\border.ebn"
        .Add 5,"CP:4 1 1 -1 -1"
    End With
```

```
    .GetOcx().BackColorPanels = &H5000000
    .GetOcx().BackColor = &H8000000f
    .Format = "1/2,(24;5/6)"
    With .get_Panel(1)
        .ControlID = "MSCAL.Calendar"
        With .Object
            .MonthLength = 0
            .BackColor = 16777215
        End With
    End With
    With .get_Panel(5)
        .Text = "<fgcolor=FFFFFF><img>1</img>75%</fgcolor>"
        .Alignment = EXSTATUSBARLib.TextAlignEnum.exAlignMiddleLeft
        .Percent = 75
        .Transparency = 35
        .Offset = "4 2 -4 -2"
    End With
    .EndUpdate
End With
```

The following **C++** sample shows "How do I access the properties and the methods of an inner ActiveX control to a panel":

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

    #import "D:\\Exontrol\\ExStatusBar\\project\\Demo\\ExStatusBar.dll"
    using namespace EXSTATUSBARLib;
*/
EXSTATUSBARLib::IStatusBarPtr spStatusBar1 = GetDlgItem(IDC_STATUSBAR1)-
>GetControlUnknown();
spStatusBar1->BeginUpdate();
spStatusBar1-
>Images(_bstr_t("gBJJgBAIDAAGAAEAAQhYAf8Pf4hh0QihCJo2AEZjQAjEZFEaIEaEEaAIAkcbk
 +
```

```
"/oFBoVDolFo1HpFJpVLplNp1PqFRqVTqlVq1XrFZrVbrldr1fsFhsVjslls1ntFptVrtltt1vuFxuVzul
 +
"/wGBwWDwmFw2HxGJxWLxmNx0xiFdyOTh8Tf9ZymXx+QytcyNgz8r0OblWjyWds+m0ka1
 +
"x3GO4NV3WeyvD2XJ5XL5nN51aiw+lfSj0gkUkAEllHanHI5j/cHg8EZf7w8vl8j4f/qfEZeB09/v
 +
"NAOAEAwCjMBwFAEDwJBMDwLBYAP2/8Hv8/gAGAD8LQs9w/nhDY/oygIA=");
EXSTATUSBARLib::IAppearancePtr var_Appearance = spStatusBar1->GetVisualAppearance();
   var_Appearance->Add(4,"c:\\exontrol\\images\\border.ebn");
   var_Appearance->Add(5,"CP:4 1 1 -1 -1");
spStatusBar1->PutBackColorPanels(83886080);
spStatusBar1->PutBackColor(-2147483633);
spStatusBar1->PutFormat(L"1/2,(24;5/6)");
EXSTATUSBARLib::IPanelPtr var_Panel = spStatusBar1->GetPanel(long(1));
   var_Panel->PutControlID(L"MSCAL.Calendar");
   /*
   Copy and paste the following directives to your header file as
   it defines the namespace 'MSACAL' for the library: 'Microsoft Calendar Control 9.0'

   #import "C:\\PROGRA~1\\MICROS~2\\Office\\MSCAL.OCX"
*/
   MSACAL::ICalendarPtr var_Calendar = ((MSACAL::ICalendarPtr)(var_Panel->GetObject()));
      var_Calendar->PutMonthLength(0);
      var_Calendar->PutBackColor(16777215);
EXSTATUSBARLib::IPanelPtr var_Panel1 = spStatusBar1->GetPanel(long(5));
   var_Panel1->PutText(L"<fgcolor=FFFFFF><img>1</img>75%</fgcolor>");
   var_Panel1->PutAlignment(EXSTATUSBARLib::exAlignMiddleLeft);
   var_Panel1->PutPercent(75);
   var_Panel1->PutTransparency(35);
   var_Panel1->PutOffset(L"4 2 -4 -2");
spStatusBar1->EndUpdate();
```

The following **C#** sample shows "How do I access the properties and the methods of an inner ActiveX control to a panel":

```
axStatusBar1.BeginUpdate();
axStatusBar1.Images("gBJJgBAIDAAGAAEAAQhYAf8Pf4hh0QihCJo2AEZjQAjEZFEaIEaEEaAIA
 +
"/oFBoVDolFo1HpFJpVLplNp1PqFRqVTqlVq1XrFZrVbrldr1fsFhsVjslls1ntFptVrtltt1vuFxuVzul
 +
"/wGBwWDwmFw2HxGJxWLxmNx0xiFdyOTh8Tf9ZymXx+QytcyNgz8r0OblWjyWds+m0ka1
 +
"x3GO4NV3WeyvD2XJ5XL5nN51aiw+lfSj0gkUkAEllHanHI5j/cHg8EZf7w8vl8j4f/qfEZeB09/v
 +
"NAOAEAwCjMBwFAEDwJBMDwLBYAP2/8Hv8/gAGAD8LQs9w/nhDY/oygIA=");
EXSTATUSBARLib.Appearance var_Appearance = axStatusBar1.VisualAppearance;
   var_Appearance.Add(4,"c:\\exontrol\\images\\border.ebn");
   var_Appearance.Add(5,"CP:4 1 1 -1 -1");
(axStatusBar1.GetOcx() as EXSTATUSBARLib.StatusBar).BackColorPanels = 0x5000000;
(axStatusBar1.GetOcx() as EXSTATUSBARLib.StatusBar).BackColor = 0x8000000f;
axStatusBar1.Format = "1/2,(24;5/6)";
EXSTATUSBARLib.Panel var_Panel = axStatusBar1.get_Panel(1);
   var_Panel.ControlID = "MSCAL.Calendar";
   // Add 'Microsoft Calendar Control 9.0' reference to your project.
   MSACAL.Calendar var_Calendar = (var_Panel.Object as MSACAL.Calendar);
      var_Calendar.MonthLength = 0;
      var_Calendar.BackColor = 16777215;
EXSTATUSBARLib.Panel var_Panel1 = axStatusBar1.get_Panel(5);
   var_Panel1.Text = "<fgcolor=FFFFFF><img>1</img>75%</fgcolor>";
   var_Panel1.Alignment = EXSTATUSBARLib.TextAlignEnum.exAlignMiddleLeft;
   var_Panel1.Percent = 75;
   var_Panel1.Transparency = 35;
   var_Panel1.Offset = "4 2 -4 -2";
axStatusBar1.EndUpdate();
```

The following **VFP** sample shows "How do I access the properties and the methods of an inner ActiveX control to a panel":

```
with thisform.StatusBar1
   .BeginUpdate
   var_s =
"gBJJgBAIDAAGAAEAAQhYAf8Pf4hh0QihCJo2AEZjQAjEZFEaIEaEEaAIAkcbk0oIUrlktl0vmExn
```

```
    var_s = var_s +
"oFBoVDolFo1HpFJpVLplNp1PqFRqVTqlVq1XrFZrVbrldr1fsFhsVjslls1ntFptVrtltt1vuFxuVzul1

    var_s = var_s +
"wGBwWDwmFw2HxGJxWLxmNx0xiFdyOTh8Tf9ZymXx+QytcyNgz8r0OblWjyWds+m0ka1V

    var_s = var_s +
"3GO4NV3WeyvD2XJ5XL5nN51aiw+lfSj0gkUkAEllHanHI5j/cHg8EZf7w8vl8j4f/qfEZeB09/vj

    var_s = var_s +
"AOAEAwCjMBwFAEDwJBMDwLBYAP2/8Hv8/gAGAD8LQs9w/nhDY/oygIA="
    .Images(var_s)
    with .VisualAppearance
        .Add(4,"c:\exontrol\images\border.ebn")
        .Add(5,"CP:4 1 1 -1 -1")
    endwith
    .BackColorPanels = 83886080
    .BackColor = -2147483633
    .Format = "1/2,(24;5/6)"
    with .Panel(1)
        .ControlID = "MSCAL.Calendar"
        with .Object
            .MonthLength = 0
            .BackColor = 16777215
        endwith
    endwith
    with .Panel(5)
        .Text = "<fgcolor=FFFFFF><img>1</img>75%</fgcolor>"
        .Alignment = 16
        .Percent = 75
        .Transparency = 35
        .Offset = "4 2 -4 -2"
    endwith
    .EndUpdate
endwith
```

property Panel.Offset as String

Specifies the offset to apply when text is being diplayed.

Type	Description
String	A string expression that indicates the padding of the caption in the panel. The list should be as "l t b r" where the l indicates the padding to left, t indicates the padding to top, and so on. For instance, "2 2 -2 -2" indicates that the text is padded 2 pixels from all sides.

Use the Offset property to add extra padding to the text being displayed in panel. Use the Text property to assign a caption to a panel. Use the OffsetPercent property to specify the padding to display the progress-bar inside the panel.

The following **VB** sample shows "How do I control the padding on the left, top or other sides":

```
With StatusBar1
   .BeginUpdate
   .Debug = True
   .VisualAppearance.Add 4,"c:\exontrol\images\border.ebn"
   .VisualAppearance.Add 5,"CP:4 1 1 -1 -1"
   .BackColorPanels = 83886080
   .BackColor = -2147483633
   .Format = "1/2/(3,4)"
   With .Panel(3)
      .Text = "Arrange the panels as you want using CRD strings"
      .Alignment = exAlignTopLeft
      .ToolTipText = .Text
      .Offset = "10 10 -10 -10"
   End With
   .EndUpdate
End With
```

The following **VB.NET** sample shows "How do I control the padding on the left, top or other sides":

```
With AxStatusBar1
   .BeginUpdate
```

```
    .Debug = True
    .VisualAppearance.Add 4,"c:\exontrol\images\border.ebn"
    .VisualAppearance.Add 5,"CP:4 1 1 -1 -1"
    .GetOcx().BackColorPanels = &H5000000
    .GetOcx().BackColor = &H8000000f
    .Format = "1/2/(3,4)"
    With .get_Panel(3)
        .Text = "Arrange the panels as you want using CRD strings"
        .Alignment = EXSTATUSBARLib.TextAlignEnum.exAlignTopLeft
        .ToolTipText = .Text
        .Offset = "10 10 -10 -10"
    End With
    .EndUpdate
End With
```

The following **C++** sample shows "How do I control the padding on the left, top or other sides":

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

    #import "C:\\WINNT\\system32\\ExStatusBar.dll"
    using namespace EXSTATUSBARLib;
*/
EXSTATUSBARLib::IStatusBarPtr spStatusBar1 = GetDlgItem(IDC_STATUSBAR1)-
>GetControlUnknown();
spStatusBar1->BeginUpdate();
spStatusBar1->PutDebug(VARIANT_TRUE);
spStatusBar1->GetVisualAppearance()->Add(4,"c:\\exontrol\\images\\border.ebn");
spStatusBar1->GetVisualAppearance()->Add(5,"CP:4 1 1 -1 -1");
spStatusBar1->PutBackColorPanels(83886080);
spStatusBar1->PutBackColor(-2147483633);
spStatusBar1->PutFormat(L"1/2/(3,4)");
EXSTATUSBARLib::IPanelPtr var_Panel = spStatusBar1->GetPanel(long(3));
    var_Panel->PutText(L"Arrange the panels as you want using CRD strings");
```

```
    var_Panel->PutAlignment(EXSTATUSBARLib::exAlignTopLeft);
    var_Panel->PutToolTipText(var_Panel->GetText());
    var_Panel->PutOffset(L"10 10 -10 -10");
spStatusBar1->EndUpdate();
```

The following **C#** sample shows "How do I control the padding on the left, top or other sides":

```
axStatusBar1.BeginUpdate();
axStatusBar1.Debug = true;
axStatusBar1.VisualAppearance.Add(4,"c:\\exontrol\\images\\border.ebn");
axStatusBar1.VisualAppearance.Add(5,"CP:4 1 1 -1 -1");
(axStatusBar1.GetOcx() as EXSTATUSBARLib.StatusBar).BackColorPanels = 0x5000000;
(axStatusBar1.GetOcx() as EXSTATUSBARLib.StatusBar).BackColor = 0x8000000f;
axStatusBar1.Format = "1/2/(3,4)";
EXSTATUSBARLib.Panel var_Panel = axStatusBar1.get_Panel(3);
    var_Panel.Text = "Arrange the panels as you want using CRD strings";
    var_Panel.Alignment = EXSTATUSBARLib.TextAlignEnum.exAlignTopLeft;
    var_Panel.ToolTipText = var_Panel.Text;
    var_Panel.Offset = "10 10 -10 -10";
axStatusBar1.EndUpdate();
```

The following **VFP** sample shows "How do I control the padding on the left, top or other sides":

```
with thisform.StatusBar1
    .BeginUpdate
    .Debug = .T.
    .VisualAppearance.Add(4,"c:\exontrol\images\border.ebn")
    .VisualAppearance.Add(5,"CP:4 1 1 -1 -1")
    .BackColorPanels = 83886080
    .BackColor = -2147483633
    .Format = "1/2/(3,4)"
    with .Panel(3)
        .Text = "Arrange the panels as you want using CRD strings"
        .Alignment = 0
        .ToolTipText = .Text
        .Offset = "10 10 -10 -10"
```

```
    endwith
    .EndUpdate
endwith
```

property Panel.OffsetPercent as String

Specifies the offset to apply when the percent bar is displayed on the panel.

Type	Description
String	A string expression that indicates the padding of the progress-bar inside the panel. The list should be as "l t b r" where the l indicates the padding to left, t indicates the padding to top, and so on. For instance, "2 2 -2 -2" indicates that the progress-bar is padded 2 pixels from all sides of the panel.

Use the OffsetPercent property to specify the padding to display the progress-bar inside the panel. Use the Percent property to display a progress bar inside the panel. Use the Offset property to add extra padding to the text being displayed in panel. Use the Text property to assign a caption to a panel. Use the BackColorPercent property to change the visual appearance of the progress bar in the panel.

The following **VB** sample shows "How can I control the padding of the percent/progressbar control":

```
With StatusBar1
   .BeginUpdate
   With .VisualAppearance
      .Add 4,"c:\exontrol\images\border.ebn"
      .Add 5,"CP:4 1 1 -1 -1"
   End With
   .BackColorPanels = 83886080
   .BackColor = -2147483633
   .Format = "1/2,(24;5/6)"
   With .Panel(5)
      .Text = "15%"
      .Percent = 15
      .OffsetPercent = "6 6 -6 -6"
   End With
   .EndUpdate
End With
```

The following **VB.NET** sample shows "How can I control the padding of the percent/progressbar control":

```
With AxStatusBar1
   .BeginUpdate
   With .VisualAppearance
      .Add 4,"c:\exontrol\images\border.ebn"
      .Add 5,"CP:4 1 1 -1 -1"
   End With
   .GetOcx().BackColorPanels = &H5000000
   .GetOcx().BackColor = &H8000000f
   .Format = "1/2,(24;5/6)"
   With .get_Panel(5)
      .Text = "15%"
      .Percent = 15
      .OffsetPercent = "6 6 -6 -6"
   End With
   .EndUpdate
End With
```

The following **C++** sample shows "How can I control the padding of the percent/progressbar control":

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

   #import "C:\\WINNT\\system32\\ExStatusBar.dll"
   using namespace EXSTATUSBARLib;
*/
EXSTATUSBARLib::IStatusBarPtr spStatusBar1 = GetDlgItem(IDC_STATUSBAR1)-
>GetControlUnknown();
spStatusBar1->BeginUpdate();
EXSTATUSBARLib::IAppearancePtr var_Appearance = spStatusBar1-
>GetVisualAppearance();
   var_Appearance->Add(4,"c:\\exontrol\\images\\border.ebn");
   var_Appearance->Add(5,"CP:4 1 1 -1 -1");
spStatusBar1->PutBackColorPanels(83886080);
spStatusBar1->PutBackColor(-2147483633);
```

```
spStatusBar1->PutFormat(L"1/2,(24;5/6)");
EXSTATUSBARLib::IPanelPtr var_Panel = spStatusBar1->GetPanel(long(5));
   var_Panel->PutText(L"15%");
   var_Panel->PutPercent(15);
   var_Panel->PutOffsetPercent(L"6 6 -6 -6");
spStatusBar1->EndUpdate();
```

The following **C#** sample shows "How can I control the padding of the percent/progressbar control":

```
axStatusBar1.BeginUpdate();
EXSTATUSBARLib.Appearance var_Appearance = axStatusBar1.VisualAppearance;
   var_Appearance.Add(4,"c:\\exontrol\\images\\border.ebn");
   var_Appearance.Add(5,"CP:4 1 1 -1 -1");
(axStatusBar1.GetOcx() as EXSTATUSBARLib.StatusBar).BackColorPanels = 0x5000000;
(axStatusBar1.GetOcx() as EXSTATUSBARLib.StatusBar).BackColor = 0x8000000f;
axStatusBar1.Format = "1/2,(24;5/6)";
EXSTATUSBARLib.Panel var_Panel = axStatusBar1.get_Panel(5);
   var_Panel.Text = "15%";
   var_Panel.Percent = 15;
   var_Panel.OffsetPercent = "6 6 -6 -6";
axStatusBar1.EndUpdate();
```

The following **VFP** sample shows "How can I control the padding of the percent/progressbar control":

```
with thisform.StatusBar1
   .BeginUpdate
   with .VisualAppearance
      .Add(4,"c:\exontrol\images\border.ebn")
      .Add(5,"CP:4 1 1 -1 -1")
   endwith
   .BackColorPanels = 83886080
   .BackColor = -2147483633
   .Format = "1/2,(24;5/6)"
   with .Panel(5)
      .Text = "15%"
      .Percent = 15
```

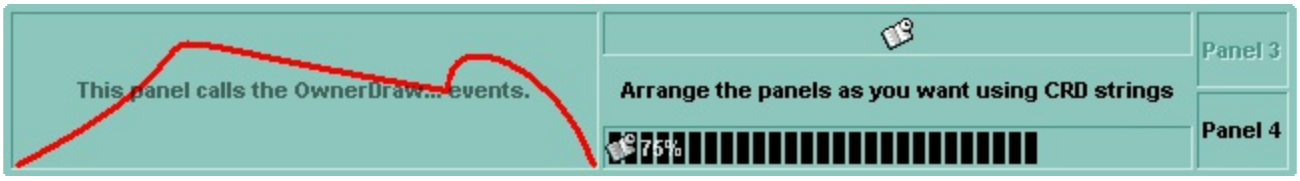
```
        .OffsetPercent = "6 6 -6 -6"
    endwith
    .EndUpdate
endwith
```

property Panel.OwnerDraw as Boolean

Specifies whether the user is responsible with painting the panel in the status bar control.

Type	Description
Boolean	A Boolean expression that specifies whether the panel is owner draw.

By default, the OwnerDraw property is False. Use the OwnerDraw property to perform your own drawing inside the panel. If the OwnerDraw property is True, the control fires the OwnerDrawStart event just before erasing and painting the panel. You can use the Transparency property to apply semi-transparent colors to the panel's content so both paintings are shown the default and your owner. The control fires the OwnerDrawEnd event when the control ends painting an owner draw panel. Use the OwnerDrawStart event to perform painting before control's default painting, and use the OwnerDrawEvent to perform your paintings after default painting is done.



The first panel in the screen shot shows a curve being drawn using the OwnerDrawStart event and still performing the default painting, as the text is painted as semi-transparent.

property Panel.Percent as Long

Specifies the percent to display the background.

Type	Description
Long	A long expression that specifies the percent of the progress bar being displayed in the panel. The value should be from 0 to 100.

Use the Percent property to indicate the percent of progress-bar value being displayed in the progress bar. The control fires the PercentChange event when the Percent value is changed. Use the BackColorPercent property to change the visual appearance of the progress bar in the panel. Use the Text property to assign a different caption to the panel. Use the Transparency property to specify a semi-transparent color for panel's caption when a progress-bar is shown. Use the BackColor property to change the visual appearance of the the panel using EBN files. Use the BackColorPanels property to assign the same visual appearance to all panels in the status bar. Use the ForeColor property to change the panel's foreground color. Use the OffsetPercent property to specify the padding to display the progress-bar inside the panel.

The following **VB** sample shows "How can I display a percent or a progress-bar inside the panel":

```
With StatusBar1
   .BeginUpdate
   .VisualAppearance.Add 4,"c:\exontrol\images\border.ebn"
   .VisualAppearance.Add 5,"CP:4 1 1 -1 -1"
   .BackColorPanels = 83886080
   .BackColor = -2147483633
   .Format = "1/2,(24;5/6)"
   With .Panel(5)
      .Text = "15%"
      .Percent = 15
   End With
   .EndUpdate
End With
```

The following **VB.NET** sample shows "How can I display a percent or a progress-bar inside

the panel":

```
With AxStatusBar1
   .BeginUpdate
   .VisualAppearance.Add 4,"c:\exontrol\images\border.ebn"
   .VisualAppearance.Add 5,"CP:4 1 1 -1 -1"
   .GetOcx().BackColorPanels = &H5000000
   .GetOcx().BackColor = &H8000000f
   .Format = "1/2,(24;5/6)"
   With .get_Panel(5)
      .Text = "15%"
      .Percent = 15
   End With
   .EndUpdate
End With
```

The following **C++** sample shows "How can I display a percent or a progress-bar inside the panel":

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

   #import "C:\\WINNT\\system32\\ExStatusBar.dll"
   using namespace EXSTATUSBARLib;
*/
EXSTATUSBARLib::IStatusBarPtr spStatusBar1 = GetDlgItem(IDC_STATUSBAR1)-
>GetControlUnknown();
spStatusBar1->BeginUpdate();
spStatusBar1->GetVisualAppearance()->Add(4,"c:\\exontrol\\images\\border.ebn");
spStatusBar1->GetVisualAppearance()->Add(5,"CP:4 1 1 -1 -1");
spStatusBar1->PutBackColorPanels(83886080);
spStatusBar1->PutBackColor(-2147483633);
spStatusBar1->PutFormat(L"1/2,(24;5/6)");
EXSTATUSBARLib::IPanelPtr var_Panel = spStatusBar1->GetPanel(long(5));
   var_Panel->PutText(L"15%");
   var_Panel->PutPercent(15);
```

```
spStatusBar1->EndUpdate();
```

The following **C#** sample shows "How can I display a percent or a progress-bar inside the panel":

```
axStatusBar1.BeginUpdate();
axStatusBar1.VisualAppearance.Add(4,"c:\\exontrol\\images\\border.ebn");
axStatusBar1.VisualAppearance.Add(5,"CP:4 1 1 -1 -1");
(axStatusBar1.GetOcx() as EXSTATUSBARLib.StatusBar).BackColorPanels = 0x5000000;
(axStatusBar1.GetOcx() as EXSTATUSBARLib.StatusBar).BackColor = 0x8000000f;
axStatusBar1.Format = "1/2,(24;5/6)";
EXSTATUSBARLib.Panel var_Panel = axStatusBar1.get_Panel(5);
   var_Panel.Text = "15%";
   var_Panel.Percent = 15;
axStatusBar1.EndUpdate();
```

The following **VFP** sample shows "How can I display a percent or a progress-bar inside the panel":

```
with thisform.StatusBar1
   .BeginUpdate
   .VisualAppearance.Add(4,"c:\exontrol\images\border.ebn")
   .VisualAppearance.Add(5,"CP:4 1 1 -1 -1")
   .BackColorPanels = 83886080
   .BackColor = -2147483633
   .Format = "1/2,(24;5/6)"
   with .Panel(5)
      .Text = "15%"
      .Percent = 15
   endwith
   .EndUpdate
endwith
```

property Panel.StrikeOut as Boolean

Specifies whether the text in the panel appears as strikeout.

Type	Description
Boolean	A Boolean expression that specifies whether the caption of the panel is shown in strikeout.

Use the StrikeOut property to show the panel's caption in strikeout. Use the Text property to assign a caption or an HTML text to a panel. Use the <s> HTML tag to show in strikeout only a portion of the caption in the panel. Use Italic the property to show the panel's caption in italic. Use the Bold property to show the panel's caption in italic. Use the Underline property to underline the panel's caption.

The following **VB** sample shows "How can I show in strikeout the caption in the panel":

```
With StatusBar1
   .BeginUpdate
   .VisualAppearance.Add 4,"c:\exontrol\images\border.ebn"
   .VisualAppearance.Add 5,"CP:4 1 1 -1 -1"
   .BackColorPanels = 83886080
   .BackColor = -2147483633
   .Format = "1/2"
   With .Panel(1)
      .Text = "Panel 1"
      .StrikeOut = True
   End With
   .Panel(2).Text = "<s>Panel</s> 2"
   .EndUpdate
End With
```

The following **VB.NET** sample shows "How can I show in strikeout the caption in the panel":

```
With AxStatusBar1
   .BeginUpdate
   .VisualAppearance.Add 4,"c:\exontrol\images\border.ebn"
   .VisualAppearance.Add 5,"CP:4 1 1 -1 -1"
   .GetOcx().BackColorPanels = &H5000000
   .GetOcx().BackColor = &H8000000f
   .Format = "1/2"
```

```
    With .get_Panel(1)
        .Text = "Panel 1"
        .StrikeOut = True
    End With
    .get_Panel(2).Text = "<s>Panel</s> 2"
    .EndUpdate
End With
```

The following **C++** sample shows "How can I show in strikeout the caption in the panel":

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

    #import "C:\\WINNT\\system32\\ExStatusBar.dll"
    using namespace EXSTATUSBARLib;
*/
EXSTATUSBARLib::IStatusBarPtr spStatusBar1 = GetDlgItem(IDC_STATUSBAR1)-
>GetControlUnknown();
spStatusBar1->BeginUpdate();
spStatusBar1->GetVisualAppearance()->Add(4,"c:\\exontrol\\images\\border.ebn");
spStatusBar1->GetVisualAppearance()->Add(5,"CP:4 1 1 -1 -1");
spStatusBar1->PutBackColorPanels(83886080);
spStatusBar1->PutBackColor(-2147483633);
spStatusBar1->PutFormat(L"1/2");
EXSTATUSBARLib::IPanelPtr var_Panel = spStatusBar1->GetPanel(long(1));
    var_Panel->PutText(L"Panel 1");
    var_Panel->PutStrikeOut(VARIANT_TRUE);
spStatusBar1->GetPanel(long(2))->PutText(L"<s>Panel</s> 2");
spStatusBar1->EndUpdate();
```

The following **C#** sample shows "How can I show in strikeout the caption in the panel":

```
axStatusBar1.BeginUpdate();
axStatusBar1.VisualAppearance.Add(4,"c:\\exontrol\\images\\border.ebn");
axStatusBar1.VisualAppearance.Add(5,"CP:4 1 1 -1 -1");
(axStatusBar1.GetOcx() as EXSTATUSBARLib.StatusBar).BackColorPanels = 0x5000000;
```

```
(axStatusBar1.GetOcx() as EXSTATUSBARLib.StatusBar).BackColor = 0x8000000f;
axStatusBar1.Format = "1/2";
EXSTATUSBARLib.Panel var_Panel = axStatusBar1.get_Panel(1);
   var_Panel.Text = "Panel 1";
   var_Panel.StrikeOut = true;
axStatusBar1.get_Panel(2).Text = "<s>Panel</s> 2";
axStatusBar1.EndUpdate();
```

The following **VFP** sample shows "How can I show in strikeout the caption in the panel":

```
with thisform.StatusBar1
   .BeginUpdate
   .VisualAppearance.Add(4,"c:\exontrol\images\border.ebn")
   .VisualAppearance.Add(5,"CP:4 1 1 -1 -1")
   .BackColorPanels = 83886080
   .BackColor = -2147483633
   .Format = "1/2"
   with .Panel(1)
      .Text = "Panel 1"
      .StrikeOut = .T.
   endwith
   .Panel(2).Text = "<s>Panel</s> 2"
   .EndUpdate
endwith
```

property Panel.Text as String

Gets or sets the text of the status bar panel.

Type	Description
String	A String expression that indicates the caption of the panel. This property supports HTML format as listed bellow.

Use the Text property to assign an HTML caption to a panel. **Use the Format property to add and arrange panels in your status bar control.** Use the Panel property to access a Panel object. Use the Debug property to display the identifiers of the panels in the status bar. Use the Enabled property to enable or disable a panel. Use the ToolTipText property to specify the tooltip being shown when the cursor hovers the panel. Use the Image property to assign a single icon to a panel, or use the <img> HTML tag to assign multiple icons or custom size pictures to a panel. Use the Images method to assign a list of icons to your status bar. Use the Percent property to display a progress bar inside the panel. Use the Transparency property to apply semi-transparent color or use the ForeColor property to specify the panel's foreground color. Use the ControlID property to display an ActiveX control inside the panel.

Use the Offset property to apply extra padding to your text inside the panel.

The Text property supports the following HTML tags:

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

with a specified background color. The rr/gg/bb represents the red/green/blue values of the color in hexa values.

- **<solidline rrggbb> ... </solidline>** or <solidline=rrggbb> ... </solidline> draws a solid-line on the bottom side of the current text-line, of specified RGB color. The <solidline> ... </solidline> draws a black solid-line on the bottom side of the current text-line. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<dotline rrggbb> ... </dotline>** or <dotline=rrggbb> ... </dotline> draws a dot-line on the bottom side of the current text-line, of specified RGB color. The <dotline> ... </dotline> draws a black dot-line on the bottom side of the current text-line. The rr/gg/bb represents the red/green/blue values of the color in hexa values.
- **<upline> ... </upline>** draws the line on the top side of the current text-line (requires <solidline> or <dotline>).
- **<r>** right aligns the text
- **<c>** centers the text
- **
** forces a line-break
- **<img>number[:width]</img>** inserts an icon inside the text. The number indicates the index of the icon being inserted. Use the Images method to assign a list of icons to your chart. The last 7 bits in the high significant byte of the number expression indicates the identifier of the skin being used to paint the object. Use the Add method to add new skins to the control. If you need to remove the skin appearance from a part of the control you need to reset the last 7 bits in the high significant byte of the color being applied to the part. The width is optional and indicates the width of the icon being inserted. Using the width option you can overwrite multiple icons getting a nice effect. By default, if the width field is missing, the width is 18 pixels.
- **<img>key[:width]</img>** inserts a custom size picture into the text being previously loaded using the HTMLPicture property. The Key parameter indicates the key of the picture being displayed. The Width parameter indicates a custom size, if you require to stretch the picture, else the original size of the picture is used.
- **&** glyph characters as **&** (&), **<** (<), **>** (>), **&qout;** (") and **&#number;** (the character with specified code), For instance, the € displays the EUR character. The **&** ampersand is only recognized as markup when it is followed by a known letter or a #character and a digit. For instance if you want to display bold in HTML caption you can use bold
- **<off offset> ... </off>** defines the vertical offset to display the text/element. The offset parameter defines the offset to display the element. This tag is inheritable, so the offset is keep while the associated </off> tag is found. You can use the <off offset> HTML tag in combination with the to define a smaller or a larger font to be displayed. For instance: "Text with <**off** 6>subscript" displays the text such as: Text with $_{\text{subscript}}$ The "Text with <**off** -6>superscript" displays the text such as: Text with ^{subscript}
- **<gra rrggbb;mode;blend> ... </gra>** defines a gradient text. The text color or <fgcolor> defines the starting gradient color, while the rr/gg/bb represents the

red/green/blue values of the ending color, 808080 if missing as gray. The mode is a value between 0 and 4, 1 if missing, and blend could be 0 or 1, 0 if missing. The HTML tag can be used to define the height of the font. Any of the rrggbb, mode or blend field may not be specified. The <gra> with no fields, shows a vertical gradient color from the current text color to gray (808080). For instance the "<**gra** FFFFFF;1;1>gradient-center</**gra**>" generates the following picture:

gradient-center

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

outlined

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

shadow

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

outline anti-aliasing

The following **VB** sample shows "How do I assign a caption or a text to a panel":

```
With StatusBar1
   .BeginUpdate
   .VisualAppearance.Add 4,"c:\exontrol\images\border.ebn"
   .VisualAppearance.Add 5,"CP:4 1 1 -1 -1"
   .BackColorPanels = 83886080
   .BackColor = -2147483633
   .Format = "1/2"
   .Panel(1).Text = "Panel 1"
```

```
    .Panel(2).Text = "Panel 2"
    .EndUpdate
End With
```

The following **VB.NET** sample shows "How do I assign a caption or a text to a panel":

```
With AxStatusBar1
    .BeginUpdate
    .VisualAppearance.Add 4,"c:\exontrol\images\border.ebn"
    .VisualAppearance.Add 5,"CP:4 1 1 -1 -1"
    .GetOcx().BackColorPanels = &H5000000
    .GetOcx().BackColor = &H8000000f
    .Format = "1/2"
    .get_Panel(1).Text = "Panel 1"
    .get_Panel(2).Text = "Panel 2"
    .EndUpdate
End With
```

The following **C++** sample shows "How do I assign a caption or a text to a panel":

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

    #import "C:\\WINNT\\system32\\ExStatusBar.dll"
    using namespace EXSTATUSBARLib;
*/
EXSTATUSBARLib::IStatusBarPtr spStatusBar1 = GetDlgItem(IDC_STATUSBAR1)-
>GetControlUnknown();
spStatusBar1->BeginUpdate();
spStatusBar1->GetVisualAppearance()->Add(4,"c:\\exontrol\\images\\border.ebn");
spStatusBar1->GetVisualAppearance()->Add(5,"CP:4 1 1 -1 -1");
spStatusBar1->PutBackColorPanels(83886080);
spStatusBar1->PutBackColor(-2147483633);
spStatusBar1->PutFormat(L"1/2");
spStatusBar1->GetPanel(long(1))->PutText(L"Panel 1");
spStatusBar1->GetPanel(long(2))->PutText(L"Panel 2");
```

```
spStatusBar1->EndUpdate();
```

The following **C#** sample shows "How do I assign a caption or a text to a panel":

```
axStatusBar1.BeginUpdate();
axStatusBar1.VisualAppearance.Add(4,"c:\\exontrol\\images\\border.ebn");
axStatusBar1.VisualAppearance.Add(5,"CP:4 1 1 -1 -1");
(axStatusBar1.GetOcx() as EXSTATUSBARLib.StatusBar).BackColorPanels = 0x5000000;
(axStatusBar1.GetOcx() as EXSTATUSBARLib.StatusBar).BackColor = 0x8000000f;
axStatusBar1.Format = "1/2";
axStatusBar1.get_Panel(1).Text = "Panel 1";
axStatusBar1.get_Panel(2).Text = "Panel 2";
axStatusBar1.EndUpdate();
```

The following **VFP** sample shows "How do I assign a caption or a text to a panel":

```
with thisform.StatusBar1
   .BeginUpdate
   .VisualAppearance.Add(4,"c:\exontrol\images\border.ebn")
   .VisualAppearance.Add(5,"CP:4 1 1 -1 -1")
   .BackColorPanels = 83886080
   .BackColor = -2147483633
   .Format = "1/2"
   .Panel(1).Text = "Panel 1"
   .Panel(2).Text = "Panel 2"
   .EndUpdate
endwith
```

property Panel.ToolTipText as String

Gets or sets ToolTip text associated with the status bar panel.

Type	Description
String	A string expression that specifies the tooltip text being displayed when the cursor hovers the panel. This property does support HTML format as listed bellow.

By default, the ToolTipText property is empty. Use the ToolTipText property to assign a tooltip being displayed when the cursor hovers the panel. Use the ToolTipTitle property to assign a title to the panel's tooltip. The control shows the tooltip only if ToolTipText property or ToolTipTitle property is not empty. Use the ShowToolTip method to programmatically display your tooltip. Use the ToolTipDelay property to specify the time in ms that passes before the ToolTip appears. Use the ToolTipPopDelay property to specify 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, in pixels.

The tooltip supports the following HTML tags:

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

... </solidline> draws a black solid-line on the bottom side of the current text-line. The rr/gg/bb represents the red/green/blue values of the color in hexa values.

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

color from the current text color to gray (808080). For instance the "<**gra** FFFFFF;1;1>gradient-center</**gra**>" generates the following picture:

gradient-center

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

outlined

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

shadow

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

outline anti-aliasing

property Panel.ToolTipTitle as String

Gets or sets ToolTip title associated with the status bar panel.

Type	Description
String	A string expression that specifies the tooltip's title. The tooltip shows up when the cursor hovers the panel. This property does not support HTML format.

By default, the ToolTipTitle property is empty. Use the ToolTipTitle property to assign a title to the panel's tooltip. Use the ToolTipText property to assign a tooltip being displayed when the cursor hovers the panel. The control shows the tooltip only if ToolTipText property or ToolTipTitle property is not empty. Use the ShowToolTip method to programmatically display your tooltip. Use the ToolTipDelay property to specify the time in ms that passes before the ToolTip appears. Use the ToolTipPopDelay property to specify 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, in pixels.

property Panel.Transparency as Long

Specifies the transparency to display the text in the panel.

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

By default, the Transparency property is 0, which means that the panel's caption is opaque. The Transparency property have effect if you perform some owner draw during the OwnerDrawStart event, or you are displaying a progress-bar and the text in the same panel. Use the Text property to assign a caption to a panel. Use the Percent property to show a progress-bar inside the panel.

The icon and the 45% are shown using 50% transparency.

The following **VB** sample shows "How can I show the percent value over the progress bar using a semi-transparent color":

```
With StatusBar1
   .BeginUpdate
   .Images
"gBJJgBAIDAAGAAEAAQhYAf8Pf4hh0QihCJo2AEZjQAjEZFEaIEaEEaAIAkcbk0olUrlktl0vmExn
 & _
"/oFBoVDolFo1HpFJpVLplNp1PqFRqVTqlVq1XrFZrVbrldr1fsFhsVjslls1ntFptVrtltt1vuFxuVzul
 & _
"/wGBwWDwmFw2HxGJxWLxmNx0xiFdyOTh8Tf9ZymXx+QytcyNgz8r0OblWjyWds+m0ka1
 & _
"x3GO4NV3WeyvD2XJ5XL5nN51aiw+lfSj0gkUkAEllHanHI5j/cHg8EZf7w8vl8j4f/qfEZeB09/v
 & _
"NAOAEAwCjMBwFAEDwJBMDwLBYAP2/8Hv8/gAGAD8LQs9w/nhDY/oygIA="
   With .VisualAppearance
      .Add 4,"c:\exontrol\images\border.ebn"
      .Add 5,"CP:4 1 1 -1 -1"
   End With
   .BackColorPanels = 83886080
```

```
    .BackColor = -2147483633
    .Format = "1/2,(24;5/6)"
    With .Panel(5)
        .Text = "<fgcolor=FFFFFF><img>1</img>75%</fgcolor>"
        .Alignment = exAlignMiddleLeft
        .Percent = 75
        .Transparency = 35
        .Offset = "4 2 -4 -2"
    End With
    .EndUpdate
End With
```

The following **VB.NET** sample shows "How can I show the percent value over the progress bar using a semi-transparent color":

```
With AxStatusBar1
    .BeginUpdate
    .Images
"gBJJgBAIDAAGAAEAAQhYAf8Pf4hh0QihCJo2AEZjQAjEZFEaIEaEEaAIAkcbk0olUrlktl0vmExn
 & _
"/oFBoVDolFo1HpFJpVLplNp1PqFRqVTqlVq1XrFZrVbrldr1fsFhsVjslls1ntFptVrtltt1vuFxuVzul
 & _
"/wGBwWDwmFw2HxGJxWLxmNx0xiFdyOTh8Tf9ZymXx+QytcyNgz8r0OblWjyWds+m0ka1
 & _
"x3GO4NV3WeyvD2XJ5XL5nN51aiw+lfSj0gkUkAEllHanHI5j/cHg8EZf7w8vl8j4f/qfEZeB09/v
 & _
"NAOAEAwCjMBwFAEDwJBMDwLBYAP2/8Hv8/gAGAD8LQs9w/nhDY/oygIA="
    With .VisualAppearance
        .Add 4,"c:\exontrol\images\border.ebn"
        .Add 5,"CP:4 1 1 -1 -1"
    End With
    .GetOcx().BackColorPanels = &H5000000
    .GetOcx().BackColor = &H8000000f
    .Format = "1/2,(24;5/6)"
    With .get_Panel(5)
        .Text = "<fgcolor=FFFFFF><img>1</img>75%</fgcolor>"
        .Alignment = EXSTATUSBARLib.TextAlignEnum.exAlignMiddleLeft
```

```
        .Percent = 75
        .Transparency = 35
        .Offset = "4 2 -4 -2"
    End With
    .EndUpdate
End With
```

The following **C++** sample shows "How can I show the percent value over the progress bar using a semi-transparent color":

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

    #import "C:\\WINNT\\system32\\ExStatusBar.dll"
    using namespace EXSTATUSBARLib;
*/
EXSTATUSBARLib::IStatusBarPtr spStatusBar1 = GetDlgItem(IDC_STATUSBAR1)-
>GetControlUnknown();
spStatusBar1->BeginUpdate();
spStatusBar1-
>Images(_bstr_t("gBJJgBAIDAAGAAEAAQhYAf8Pf4hh0QihCJo2AEZjQAjEZFEaIEaEEaAIAkcbk
 +
"/oFBoVDolFo1HpFJpVLplNp1PqFRqVTqlVq1XrFZrVbrldr1fsFhsVjslls1ntFptVrtltt1vuFxuVzul
 +
"/wGBwWDwmFw2HxGJxWLxmNx0xiFdyOTh8Tf9ZymXx+QytcyNgz8r0OblWjyWds+m0ka1
 +
"x3GO4NV3WeyvD2XJ5XL5nN51aiw+lfSj0gkUkAEllHanHI5j/cHg8EZf7w8vl8j4f/qfEZeB09/v
 +
"NAOAEAwCjMBwFAEDwJBMDwLBYAP2/8Hv8/gAGAD8LQs9w/nhDY/oygIA=");
EXSTATUSBARLib::IAppearancePtr var_Appearance = spStatusBar1-
>GetVisualAppearance();
    var_Appearance->Add(4,"c:\\exontrol\\images\\border.ebn");
    var_Appearance->Add(5,"CP:4 1 1 -1 -1");
spStatusBar1->PutBackColorPanels(83886080);
spStatusBar1->PutBackColor(-2147483633);
```

```
spStatusBar1->PutFormat(L"1/2,(24;5/6)");
EXSTATUSBARLib::IPanelPtr var_Panel = spStatusBar1->GetPanel(long(5));
   var_Panel->PutText(L"<fgcolor=FFFFFF><img>1</img>75%</fgcolor>");
   var_Panel->PutAlignment(EXSTATUSBARLib::exAlignMiddleLeft);
   var_Panel->PutPercent(75);
   var_Panel->PutTransparency(35);
   var_Panel->PutOffset(L"4 2 -4 -2");
spStatusBar1->EndUpdate();
```

The following **C#** sample shows "How can I show the percent value over the progress bar using a semi-transparent color":

```
axStatusBar1.BeginUpdate();
axStatusBar1.Images("gBJJgBAIDAAGAAEAAQhYAf8Pf4hh0QihCJo2AEZjQAjEZFEaIEaEEaAIA
 +
"/oFBoVDolFo1HpFJpVLplNp1PqFRqVTqlVq1XrFZrVbrldr1fsFhsVjslls1ntFptVrtltt1vuFxuVzul
 +
"/wGBwWDwmFw2HxGJxWLxmNx0xiFdyOTh8Tf9ZymXx+QytcyNgz8r0OblWjyWds+m0ka1
 +
"x3GO4NV3WeyvD2XJ5XL5nN51aiw+lfSj0gkUkAEllHanHI5j/cHg8EZf7w8vl8j4f/qfEZeB09/v
 +
"NAOAEAwCjMBwFAEDwJBMDwLBYAP2/8Hv8/gAGAD8LQs9w/nhDY/oygIA=");
EXSTATUSBARLib.Appearance var_Appearance = axStatusBar1.VisualAppearance;
   var_Appearance.Add(4,"c:\\exontrol\\images\\border.ebn");
   var_Appearance.Add(5,"CP:4 1 1 -1 -1");
(axStatusBar1.GetOcx() as EXSTATUSBARLib.StatusBar).BackColorPanels = 0x5000000;
(axStatusBar1.GetOcx() as EXSTATUSBARLib.StatusBar).BackColor = 0x8000000f;
axStatusBar1.Format = "1/2,(24;5/6)";
EXSTATUSBARLib.Panel var_Panel = axStatusBar1.get_Panel(5);
   var_Panel.Text = "<fgcolor=FFFFFF><img>1</img>75%</fgcolor>";
   var_Panel.Alignment = EXSTATUSBARLib.TextAlignEnum.exAlignMiddleLeft;
   var_Panel.Percent = 75;
   var_Panel.Transparency = 35;
   var_Panel.Offset = "4 2 -4 -2";
axStatusBar1.EndUpdate();
```

The following **VFP** sample shows "How can I show the percent value over the progress bar using a semi-transparent color":

```
with thisform.StatusBar1
   .BeginUpdate
   var_s =
"gBJJgBAIDAAGAAEAAQhYAf8Pf4hh0QihCJo2AEZjQAjEZFEaIEaEEaAIAkcbk0olUrlktl0vmExn

   var_s = var_s +
"oFBoVDolFo1HpFJpVLplNp1PqFRqVTqlVq1XrFZrVbrldr1fsFhsVjslls1ntFptVrtltt1vuFxuVzul1

   var_s = var_s +
"wGBwWDwmFw2HxGJxWLxmNx0xiFdyOTh8Tf9ZymXx+QytcyNgz8r0OblWjyWds+m0ka1V

   var_s = var_s +
"3GO4NV3WeyvD2XJ5XL5nN51aiw+lfSj0gkUkAEllHanHI5j/cHg8EZf7w8vl8j4f/qfEZeB09/vj

   var_s = var_s +
"AOAEAwCjMBwFAEDwJBMDwLBYAP2/8Hv8/gAGAD8LQs9w/nhDY/oygIA="
   .Images(var_s)
   with .VisualAppearance
      .Add(4,"c:\exontrol\images\border.ebn")
      .Add(5,"CP:4 1 1 -1 -1")
   endwith
   .BackColorPanels = 83886080
   .BackColor = -2147483633
   .Format = "1/2,(24;5/6)"
   with .Panel(5)
      .Text = "<fgcolor=FFFFFF><img>1</img>75%</fgcolor>"
      .Alignment = 16
      .Percent = 75
      .Transparency = 35
      .Offset = "4 2 -4 -2"
   endwith
   .EndUpdate
endwith
```

property Panel.Underline as Boolean

Specifies whether the text in the panel appears as underlined.

Type	Description
Boolean	A Boolean expression that specifies whether the caption of the panel is underlined.

Use the Underline property to underline the panel's caption. Use the Text property to assign a caption or an HTML text to a panel. Use the <u> or <a> HTML tag to underline a portion of the caption in the panel. Use Italic the property to show the panel's caption in italic. Use the Bold property to show the panel's caption in italic. Use the StrikeOut property to show the panel's caption in strikeout.

The following **VB** sample shows "How can I underline the caption in the panel":

```
With StatusBar1
   .BeginUpdate
   .VisualAppearance.Add 4,"c:\exontrol\images\border.ebn"
   .VisualAppearance.Add 5,"CP:4 1 1 -1 -1"
   .BackColorPanels = 83886080
   .BackColor = -2147483633
   .Format = "1/2"
   With .Panel(1)
      .Text = "Panel 1"
      .Underline = True
   End With
   .Panel(2).Text = "<u>Panel</u> 2"
   .EndUpdate
End With
```

The following **VB.NET** sample shows "How can I underline the caption in the panel":

```
With AxStatusBar1
   .BeginUpdate
   .VisualAppearance.Add 4,"c:\exontrol\images\border.ebn"
   .VisualAppearance.Add 5,"CP:4 1 1 -1 -1"
   .GetOcx().BackColorPanels = &H5000000
   .GetOcx().BackColor = &H8000000f
   .Format = "1/2"
```

```
    With .get_Panel(1)
        .Text = "Panel 1"
        .Underline = True
    End With
    .get_Panel(2).Text = "<u>Panel</u> 2"
    .EndUpdate
End With
```

The following **C++** sample shows "How can I underline the caption in the panel":

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

    #import "C:\\WINNT\\system32\\ExStatusBar.dll"
    using namespace EXSTATUSBARLib;
*/
EXSTATUSBARLib::IStatusBarPtr spStatusBar1 = GetDlgItem(IDC_STATUSBAR1)-
>GetControlUnknown();
spStatusBar1->BeginUpdate();
spStatusBar1->GetVisualAppearance()->Add(4,"c:\\exontrol\\images\\border.ebn");
spStatusBar1->GetVisualAppearance()->Add(5,"CP:4 1 1 -1 -1");
spStatusBar1->PutBackColorPanels(83886080);
spStatusBar1->PutBackColor(-2147483633);
spStatusBar1->PutFormat(L"1/2");
EXSTATUSBARLib::IPanelPtr var_Panel = spStatusBar1->GetPanel(long(1));
    var_Panel->PutText(L"Panel 1");
    var_Panel->PutUnderline(VARIANT_TRUE);
spStatusBar1->GetPanel(long(2))->PutText(L"<u>Panel</u> 2");
spStatusBar1->EndUpdate();
```

The following **C#** sample shows "How can I underline the caption in the panel":

```
axStatusBar1.BeginUpdate();
axStatusBar1.VisualAppearance.Add(4,"c:\\exontrol\\images\\border.ebn");
axStatusBar1.VisualAppearance.Add(5,"CP:4 1 1 -1 -1");
(axStatusBar1.GetOcx() as EXSTATUSBARLib.StatusBar).BackColorPanels = 0x5000000;
```

```
(axStatusBar1.GetOcx() as EXSTATUSBARLib.StatusBar).BackColor = 0x8000000f;
axStatusBar1.Format = "1/2";
EXSTATUSBARLib.Panel var_Panel = axStatusBar1.get_Panel(1);
   var_Panel.Text = "Panel 1";
   var_Panel.Underline = true;
axStatusBar1.get_Panel(2).Text = "<u>Panel</u> 2";
axStatusBar1.EndUpdate();
```

The following **VFP** sample shows "How can I underline the caption in the panel":

```
with thisform.StatusBar1
   .BeginUpdate
   .VisualAppearance.Add(4,"c:\exontrol\images\border.ebn")
   .VisualAppearance.Add(5,"CP:4 1 1 -1 -1")
   .BackColorPanels = 83886080
   .BackColor = -2147483633
   .Format = "1/2"
   with .Panel(1)
      .Text = "Panel 1"
      .Underline = .T.
   endwith
   .Panel(2).Text = "<u>Panel</u> 2"
   .EndUpdate
endwith
```

property Panel.UserData as Variant

Associates an extra data to the panel.

Type	Description
Variant	A Variant expression that specifies any extra data associated to a panel. If could be a number, a string, a date, a reference to an object or anything that a VARIANT expression could hold.

Use the UserData property to assign your extra data to a panel. Use the Text property to assign a caption to a panel.

The following **VB** sample shows "How can I assign an extra data to my panel":

```
With StatusBar1
   .BeginUpdate
   .VisualAppearance.Add 4,"c:\exontrol\images\border.ebn"
   .VisualAppearance.Add 5,"CP:4 1 1 -1 -1"
   .BackColorPanels = 83886080
   .BackColor = -2147483633
   .Format = "1/2,(5/6/7/8)"
   With .Panel(5)
      .Text = "UserData"
      .UserData = "this is just some extra data associated to the panel"
      .ToolTipText = .UserData
   End With
   .EndUpdate
End With
```

The following **VB.NET** sample shows "How can I assign an extra data to my panel":

```
With AxStatusBar1
   .BeginUpdate
   .VisualAppearance.Add 4,"c:\exontrol\images\border.ebn"
   .VisualAppearance.Add 5,"CP:4 1 1 -1 -1"
   .GetOcx().BackColorPanels = &H5000000
   .GetOcx().BackColor = &H8000000f
   .Format = "1/2,(5/6/7/8)"
   With .get_Panel(5)
```

```
        .Text = "UserData"
        .UserData = "this is just some extra data associated to the panel"
        .ToolTipText = .UserData
    End With
    .EndUpdate
End With
```

The following **C++** sample shows "How can I assign an extra data to my panel":

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

    #import "C:\\WINNT\\system32\\ExStatusBar.dll"
    using namespace EXSTATUSBARLib;
*/
EXSTATUSBARLib::IStatusBarPtr spStatusBar1 = GetDlgItem(IDC_STATUSBAR1)-
>GetControlUnknown();
spStatusBar1->BeginUpdate();
spStatusBar1->GetVisualAppearance()->Add(4,"c:\\exontrol\\images\\border.ebn");
spStatusBar1->GetVisualAppearance()->Add(5,"CP:4 1 1 -1 -1");
spStatusBar1->PutBackColorPanels(83886080);
spStatusBar1->PutBackColor(-2147483633);
spStatusBar1->PutFormat(L"1/2,(5/6/7/8)");
EXSTATUSBARLib::IPanelPtr var_Panel = spStatusBar1->GetPanel(long(5));
    var_Panel->PutText(L"UserData");
    var_Panel->PutUserData("this is just some extra data associated to the panel");
    var_Panel->PutToolTipText(_bstr_t(var_Panel->GetUserData()));
spStatusBar1->EndUpdate();
```

The following **C#** sample shows "How can I assign an extra data to my panel":

```
axStatusBar1.BeginUpdate();
axStatusBar1.VisualAppearance.Add(4,"c:\\exontrol\\images\\border.ebn");
axStatusBar1.VisualAppearance.Add(5,"CP:4 1 1 -1 -1");
(axStatusBar1.GetOcx() as EXSTATUSBARLib.StatusBar).BackColorPanels = 0x5000000;
(axStatusBar1.GetOcx() as EXSTATUSBARLib.StatusBar).BackColor = 0x8000000f;
```

```
axStatusBar1.Format = "1/2,(5/6/7/8)";
EXSTATUSBARLib.Panel var_Panel = axStatusBar1.get_Panel(5);
   var_Panel.Text = "UserData";
   var_Panel.UserData = "this is just some extra data associated to the panel";
   var_Panel.ToolTipText = var_Panel.UserData.ToString();
axStatusBar1.EndUpdate();
```

The following **VFP** sample shows "How can I assign an extra data to my panel":

```
with thisform.StatusBar1
   .BeginUpdate
   .VisualAppearance.Add(4,"c:\exontrol\images\border.ebn")
   .VisualAppearance.Add(5,"CP:4 1 1 -1 -1")
   .BackColorPanels = 83886080
   .BackColor = -2147483633
   .Format = "1/2,(5/6/7/8)"
   with .Panel(5)
      .Text = "UserData"
      .UserData = "this is just some extra data associated to the panel"
      .ToolTipText = .UserData
   endwith
   .EndUpdate
endwith
```

property Panel.Width as Long

Specifies the width in pixels of the panel.

Type	Description
Long	A long expression that specifies the width of the panel, in pixels.

Use the Width property to specify the width of the panel. Use the Height property to get the height in pixels of the panel. Use the Format property to control the width and the height of the panels, using CRD strings.

The following **VB** sample shows "Is there any property to get the width/height of the panel":

```
With StatusBar1
   .BeginUpdate
   .VisualAppearance.Add 4,"c:\exontrol\images\border.ebn"
   .VisualAppearance.Add 5,"CP:4 1 1 -1 -1"
   .BackColorPanels = 83886080
   .BackColor = -2147483633
   .Format = "1/2,(5/6)"
   With .Panel(5)
      .Text = .Width
   End With
   With .Panel(6)
      .Text = .Height
   End With
   .EndUpdate
End With
```

The following **VB.NET** sample shows "Is there any property to get the width/height of the panel":

```
With AxStatusBar1
   .BeginUpdate
   .VisualAppearance.Add 4,"c:\exontrol\images\border.ebn"
   .VisualAppearance.Add 5,"CP:4 1 1 -1 -1"
   .GetOcx().BackColorPanels = &H5000000
   .GetOcx().BackColor = &H8000000f
   .Format = "1/2,(5/6)"
```

```
    With .get_Panel(5)
        .Text = .Width
    End With
    With .get_Panel(6)
        .Text = .Height
    End With
    .EndUpdate
End With
```

The following **C++** sample shows "Is there any property to get the width/height of the panel":

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

    #import "C:\\WINNT\\system32\\ExStatusBar.dll"
    using namespace EXSTATUSBARLib;
*/
EXSTATUSBARLib::IStatusBarPtr spStatusBar1 = GetDlgItem(IDC_STATUSBAR1)-
>GetControlUnknown();
spStatusBar1->BeginUpdate();
spStatusBar1->GetVisualAppearance()->Add(4,"c:\\exontrol\\images\\border.ebn");
spStatusBar1->GetVisualAppearance()->Add(5,"CP:4 1 1 -1 -1");
spStatusBar1->PutBackColorPanels(83886080);
spStatusBar1->PutBackColor(-2147483633);
spStatusBar1->PutFormat(L"1/2,(5/6)");
EXSTATUSBARLib::IPanelPtr var_Panel = spStatusBar1->GetPanel(long(5));
    var_Panel->PutText(_bstr_t(var_Panel->GetWidth()));
EXSTATUSBARLib::IPanelPtr var_Panel1 = spStatusBar1->GetPanel(long(6));
    var_Panel1->PutText(_bstr_t(var_Panel1->GetHeight()));
spStatusBar1->EndUpdate();
```

The following **C#** sample shows "Is there any property to get the width/height of the panel":

```
axStatusBar1.BeginUpdate();
axStatusBar1.VisualAppearance.Add(4,"c:\\exontrol\\images\\border.ebn");
```

```
axStatusBar1.VisualAppearance.Add(5,"CP:4 1 1 -1 -1");
(axStatusBar1.GetOcx() as EXSTATUSBARLib.StatusBar).BackColorPanels = 0x5000000;
(axStatusBar1.GetOcx() as EXSTATUSBARLib.StatusBar).BackColor = 0x8000000f;
axStatusBar1.Format = "1/2,(5/6)";
EXSTATUSBARLib.Panel var_Panel = axStatusBar1.get_Panel(5);
   var_Panel.Text = var_Panel.Width.ToString();
EXSTATUSBARLib.Panel var_Panel1 = axStatusBar1.get_Panel(6);
   var_Panel1.Text = var_Panel1.Height.ToString();
axStatusBar1.EndUpdate();
```

The following **VFP** sample shows "Is there any property to get the width/height of the panel":

```
with thisform.StatusBar1
   .BeginUpdate
   .VisualAppearance.Add(4,"c:\exontrol\images\border.ebn")
   .VisualAppearance.Add(5,"CP:4 1 1 -1 -1")
   .BackColorPanels = 83886080
   .BackColor = -2147483633
   .Format = "1/2,(5/6)"
   with .Panel(5)
      .Text = .Width
   endwith
   with .Panel(6)
      .Text = .Height
   endwith
   .EndUpdate
endwith
```

property Panel.WordWrap as Boolean

Specifies whether the text is word wrapping in the panel.

Type	Description
Boolean	A Boolean expression that specifies whether the panel's caption is shown using multiple lines or single line.

By default, the WordWrap property is True. If the WordWrap property is True, the panel's Text is wrapped in the panel.

The following **VB** sample shows "How can I display the panel using a single line":

```
With StatusBar1
   .BeginUpdate
   .Debug = True
   .VisualAppearance.Add 4,"c:\exontrol\images\border.ebn"
   .VisualAppearance.Add 5,"CP:4 1 1 -1 -1"
   .BackColorPanels = 83886080
   .BackColor = -2147483633
   .Format = "1/2/(3,4)"
   With .Panel(3)
      .Text = "Arrange the panels as you want using CRD strings"
      .ToolTipText = .Text
      .WordWrap = False
   End With
   .EndUpdate
End With
```

The following **VB.NET** sample shows "How can I display the panel using a single line":

```
With AxStatusBar1
   .BeginUpdate
   .Debug = True
   .VisualAppearance.Add 4,"c:\exontrol\images\border.ebn"
   .VisualAppearance.Add 5,"CP:4 1 1 -1 -1"
   .GetOcx().BackColorPanels = &H5000000
   .GetOcx().BackColor = &H8000000f
   .Format = "1/2/(3,4)"
   With .get_Panel(3)
```

```
        .Text = "Arrange the panels as you want using CRD strings"
        .ToolTipText = .Text
        .WordWrap = False
    End With
    .EndUpdate
End With
```

The following **C++** sample shows "How can I display the panel using a single line":

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

    #import "C:\\WINNT\\system32\\ExStatusBar.dll"
    using namespace EXSTATUSBARLib;
*/
EXSTATUSBARLib::IStatusBarPtr spStatusBar1 = GetDlgItem(IDC_STATUSBAR1)-
>GetControlUnknown();
spStatusBar1->BeginUpdate();
spStatusBar1->PutDebug(VARIANT_TRUE);
spStatusBar1->GetVisualAppearance()->Add(4,"c:\\exontrol\\images\\border.ebn");
spStatusBar1->GetVisualAppearance()->Add(5,"CP:4 1 1 -1 -1");
spStatusBar1->PutBackColorPanels(83886080);
spStatusBar1->PutBackColor(-2147483633);
spStatusBar1->PutFormat(L"1/2/(3,4)");
EXSTATUSBARLib::IPanelPtr var_Panel = spStatusBar1->GetPanel(long(3));
    var_Panel->PutText(L"Arrange the panels as you want using CRD strings");
    var_Panel->PutToolTipText(var_Panel->GetText());
    var_Panel->PutWordWrap(VARIANT_FALSE);
spStatusBar1->EndUpdate();
```

The following **C#** sample shows "How can I display the panel using a single line":

```
axStatusBar1.BeginUpdate();
axStatusBar1.Debug = true;
axStatusBar1.VisualAppearance.Add(4,"c:\\exontrol\\images\\border.ebn");
axStatusBar1.VisualAppearance.Add(5,"CP:4 1 1 -1 -1");
```

```
(axStatusBar1.GetOcx() as EXSTATUSBARLib.StatusBar).BackColorPanels = 0x5000000;
(axStatusBar1.GetOcx() as EXSTATUSBARLib.StatusBar).BackColor = 0x8000000f;
axStatusBar1.Format = "1/2/(3,4)";
EXSTATUSBARLib.Panel var_Panel = axStatusBar1.get_Panel(3);
   var_Panel.Text = "Arrange the panels as you want using CRD strings";
   var_Panel.ToolTipText = var_Panel.Text;
   var_Panel.WordWrap = false;
axStatusBar1.EndUpdate();
```

The following **VFP** sample shows "How can I display the panel using a single line":

```
with thisform.StatusBar1
   .BeginUpdate
   .Debug = .T.
   .VisualAppearance.Add(4,"c:\exontrol\images\border.ebn")
   .VisualAppearance.Add(5,"CP:4 1 1 -1 -1")
   .BackColorPanels = 83886080
   .BackColor = -2147483633
   .Format = "1/2/(3,4)"
   with .Panel(3)
      .Text = "Arrange the panels as you want using CRD strings"
      .ToolTipText = .Text
      .WordWrap = .F.
   endwith
   .EndUpdate
endwith
```

StatusBar 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: {0885027A-DF96-481F-928C-E3E3788889BA}. The object's program identifier is: "Exontrol.StatusBar". The /COM object module is: "ExStatusBar.dll"

The Exontrol's ExStatusBar component provides statusbar panels to your forms. The statusbar is a component (widget) often found at the bottom of windows in a graphical user interface. It is very frequently divided into sections, each of which shows different information. Its job is primarily to display information about the current state of its window, although some status bars have extra functionality. Usually, the status bar often called a status line in this context displays the current state of the application, and helpful keyboard shortcuts. **Use the Format property to add and arrange the panels in the status bar control. Use the Panel property to access the panels in the status bar**.

Name	Description
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.
BackColor	Specifies the control's background color.
BackColorPanels	Specifies a background color or a visual appearance applied to all panels in the status bar control.
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.
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.
ClearPanels	Clears the collection of panels in the control.
Debug	Specifies whether the control displays debug information such of identifiers of the panels.
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.
Format	Specifies the CRD format to arrange the objects inside the control.
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 Images List Control.
ImageSize	Retrieves or sets the size of icons the control displays..
Panel	Retrieves the panel in the control giving its identifier.
PanelFromPoint	Retrieves the panel from the point.
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.
ReplaceIcon	Adds a new icon, replaces an icon or clears the control's image list.
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.
TemplateResult	Gets the result of the last Template call.
TemplateResultN	Gets the result of the last Template call, as double.
TemplateResultS	Gets the result of the last Template call, as string.
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.
Version	Retrieves the control's version.
VisualAppearance	Retrieves the control's appearance.

property StatusBar.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 StatusBar1_MouseMove(Button As Integer, Shift As Integer, x As Single, y As Single)
   With StatusBar1
      .ShowToolTip .AnchorFromPoint(-1, -1), , 3
   End With
End Sub
```

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

```
Private Sub AxStatusBar1_MouseMoveEvent(ByVal sender As Object, ByVal e As AxEXSTATUSBARLib._IStatusBarEvents_MouseMoveEvent) Handles AxStatusBar1.MouseMoveEvent
   With AxStatusBar1
      .ShowToolTip(.get_AnchorFromPoint(-1, -1), "", 3, -1, -1)
   End With
```

```
End Sub
```

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

```
private void axStatusBar1_MouseMoveEvent(object sender,
AxEXSTATUSBARLib._IStatusBarEvents_MouseMoveEvent e)
{
   axStatusBar1.ShowToolTip(axStatusBar1.get_AnchorFromPoint(-1, -1), "", 3, -1, -1);
}
```

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

```
void OnMouseMoveStatusbar1(short Button, short Shift, long X, long Y)
{
   m_statusBar.ShowToolTip( m_statusBar.GetAnchorFromPoint( -1,-1), COleVariant(""),
COleVariant((long)3), COleVariant( long(-1) ), COleVariant( long(-1) ) );
}
```

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

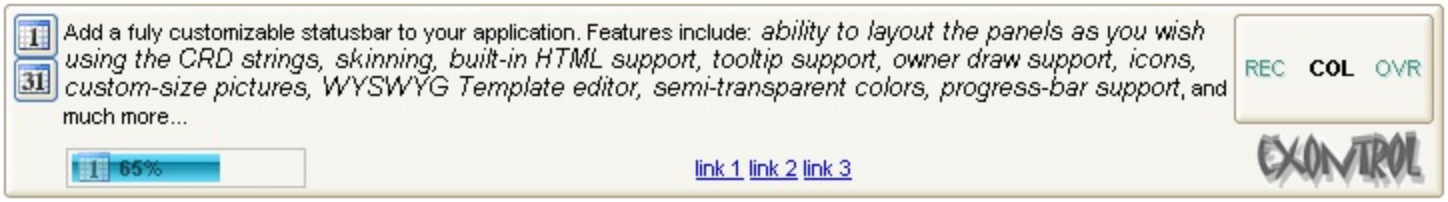
```
*** ActiveX Control Event ***
LPARAMETERS button, shift, x, y

with thisform.StatusBar1
   .ShowToolTip(.AnchorFromPoint(-1,-1), "", 3, -1, -1 )
endwith
```

property StatusBar.Appearance as AppearanceEnum

Retrieves or sets the control's appearance.

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

Use the Appearance property to specify the control's border. Use the 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 shows "How do I change the control's border, using your EBN files":

```
With StatusBar1
   .BeginUpdate
   .VisualAppearance.Add 1,"c:\exontrol\images\normal.ebn"
   .Appearance = 16777216
   .VisualAppearance.Add 4,"c:\exontrol\images\border.ebn"
   .VisualAppearance.Add 5,"CP:4 1 1 -1 -1"
   .BackColorPanels = 83886080
   .Format = "1,2,3,4,(5/6/7/8)"
   .Debug = True
```

```
    .EndUpdate
End With
```

The following **VB.NET** sample shows "How do I change the control's border, using your EBN files":

```
With AxStatusBar1
    .BeginUpdate
    .VisualAppearance.Add 1,"c:\exontrol\images\normal.ebn"
    .Appearance = 16777216
    .VisualAppearance.Add 4,"c:\exontrol\images\border.ebn"
    .VisualAppearance.Add 5,"CP:4 1 1 -1 -1"
    .GetOcx().BackColorPanels = &H5000000
    .Format = "1,2,3,4,(5/6/7/8)"
    .Debug = True
    .EndUpdate
End With
```

The following **C++** sample shows "How do I change the control's border, using your EBN files":

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

    #import "D:\\Exontrol\\ExStatusBar\\project\\Demo\\ExStatusBar.dll"
    using namespace EXSTATUSBARLib;
*/
EXSTATUSBARLib::IStatusBarPtr spStatusBar1 = GetDlgItem(IDC_STATUSBAR1)-
>GetControlUnknown();
spStatusBar1->BeginUpdate();
spStatusBar1->GetVisualAppearance()->Add(1,"c:\\exontrol\\images\\normal.ebn");
spStatusBar1->PutAppearance((EXSTATUSBARLib::AppearanceEnum)16777216);
spStatusBar1->GetVisualAppearance()->Add(4,"c:\\exontrol\\images\\border.ebn");
spStatusBar1->GetVisualAppearance()->Add(5,"CP:4 1 1 -1 -1");
spStatusBar1->PutBackColorPanels(83886080);
spStatusBar1->PutFormat(L"1,2,3,4,(5/6/7/8)");
```

```
spStatusBar1->PutDebug(VARIANT_TRUE);
spStatusBar1->EndUpdate();
```

The following **C#** sample shows "How do I change the control's border, using your EBN files":

```
axStatusBar1.BeginUpdate();
axStatusBar1.VisualAppearance.Add(1,"c:\\exontrol\\images\\normal.ebn");
axStatusBar1.Appearance = (EXSTATUSBARLib.AppearanceEnum)16777216;
axStatusBar1.VisualAppearance.Add(4,"c:\\exontrol\\images\\border.ebn");
axStatusBar1.VisualAppearance.Add(5,"CP:4 1 1 -1 -1");
(axStatusBar1.GetOcx() as EXSTATUSBARLib.StatusBar).BackColorPanels = 0x5000000;
axStatusBar1.Format = "1,2,3,4,(5/6/7/8)";
axStatusBar1.Debug = true;
axStatusBar1.EndUpdate();
```

The following **VFP** sample shows "How do I change the control's border, using your EBN files":

```
with thisform.StatusBar1
   .BeginUpdate
   .VisualAppearance.Add(1,"c:\exontrol\images\normal.ebn")
   .Appearance = 16777216
   .VisualAppearance.Add(4,"c:\exontrol\images\border.ebn")
   .VisualAppearance.Add(5,"CP:4 1 1 -1 -1")
   .BackColorPanels = 83886080
   .Format = "1,2,3,4,(5/6/7/8)"
   .Debug = .T.
   .EndUpdate
endwith
```

method StatusBar.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 StatusBar1_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 StatusBar.BackColor as Color

Specifies the control's background color.

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

Use the BackColor property to change the control's background color. Use the Picture property to layout a picture on the control's background. Use the Appearance property to change the visual appearance of the control's border. Use the BackColorPanels property to assign the same visual aspect for all panels in the status bar control. Use the ForeColor property to specify the control's foreground color.

The following **VB** sample shows "How do I change the control's background color":

```
With StatusBar1
   .BeginUpdate
   .BackColor = 13158600
   .VisualAppearance.Add 4,"c:\exontrol\images\border.ebn"
   .VisualAppearance.Add 5,"CP:4 1 1 -1 -1"
   .BackColorPanels = 83886080
   .Format = "1,2,3,4,(5/6/7/8)"
   .Debug = True
   .EndUpdate
End With
```

The following **VB.NET** sample shows "How do I change the control's background color":

```
With AxStatusBar1
   .BeginUpdate
   .BackColor = Color.FromArgb(200,200,200)
   .VisualAppearance.Add 4,"c:\exontrol\images\border.ebn"
   .VisualAppearance.Add 5,"CP:4 1 1 -1 -1"
   .GetOcx().BackColorPanels = &H5000000
   .Format = "1,2,3,4,(5/6/7/8)"
   .Debug = True
   .EndUpdate
End With
```

The following **C++** sample shows "How do I change the control's background color":

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

   #import "D:\\Exontrol\\ExStatusBar\\project\\Demo\\ExStatusBar.dll"
   using namespace EXSTATUSBARLib;
*/
EXSTATUSBARLib::IStatusBarPtr spStatusBar1 = GetDlgItem(IDC_STATUSBAR1)-
>GetControlUnknown();
spStatusBar1->BeginUpdate();
spStatusBar1->PutBackColor(13158600);
spStatusBar1->GetVisualAppearance()->Add(4,"c:\\exontrol\\images\\border.ebn");
spStatusBar1->GetVisualAppearance()->Add(5,"CP:4 1 1 -1 -1");
spStatusBar1->PutBackColorPanels(83886080);
spStatusBar1->PutFormat(L"1,2,3,4,(5/6/7/8)");
spStatusBar1->PutDebug(VARIANT_TRUE);
spStatusBar1->EndUpdate();
```

The following **C#** sample shows "How do I change the control's background color":

```
axStatusBar1.BeginUpdate();
axStatusBar1.BackColor = Color.FromArgb(200,200,200);
axStatusBar1.VisualAppearance.Add(4,"c:\\exontrol\\images\\border.ebn");
axStatusBar1.VisualAppearance.Add(5,"CP:4 1 1 -1 -1");
(axStatusBar1.GetOcx() as EXSTATUSBARLib.StatusBar).BackColorPanels = 0x5000000;
axStatusBar1.Format = "1,2,3,4,(5/6/7/8)";
axStatusBar1.Debug = true;
axStatusBar1.EndUpdate();
```

The following **VFP** sample shows "How do I change the control's background color":

```
with thisform.StatusBar1
   .BeginUpdate
   .BackColor = 13158600
   .VisualAppearance.Add(4,"c:\exontrol\images\border.ebn")
```

```
   .VisualAppearance.Add(5,"CP:4 1 1 -1 -1")
   .BackColorPanels = 83886080
   .Format = "1,2,3,4,(5/6/7/8)"
   .Debug = .T.
   .EndUpdate
endwith
```

property StatusBar.BackColorPanels as Color

Specifies a background color or a visual appearance applied to all panels in the status bar control.

Type	Description
Color	A Color expression that specifies the panels 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 BackColorPanels property to change the visual appearance of all panels in the status bar control, using EBN files. Use the BackColor property to assign a different visual appearance to a specified panel. Use the BackColor property to change the control's background color.

The following **VB** sample shows "How do I draw a border for all panels":

```
With StatusBar1
   .BeginUpdate
   .VisualAppearance.Add 4,"c:\exontrol\images\border.ebn"
   .VisualAppearance.Add 5,"CP:4 2 2 -2 -2"
   .BackColorPanels = 83886080
   .Debug = True
   .Format = "(0/1:32),2,(3/4/5)"
   .EndUpdate
End With
```

The following **VB.NET** sample shows "How do I draw a border for all panels":

```
With AxStatusBar1
   .BeginUpdate
   .VisualAppearance.Add 4,"c:\exontrol\images\border.ebn"
   .VisualAppearance.Add 5,"CP:4 2 2 -2 -2"
   .GetOcx().BackColorPanels = &H5000000
   .Debug = True
   .Format = "(0/1:32),2,(3/4/5)"
   .EndUpdate
```

```
End With
```

The following **C++** sample shows "How do I draw a border for all panels":

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

   #import "C:\\WINNT\\system32\\ExStatusBar.dll"
   using namespace EXSTATUSBARLib;
*/
EXSTATUSBARLib::IStatusBarPtr spStatusBar1 = GetDlgItem(IDC_STATUSBAR1)-
>GetControlUnknown();
spStatusBar1->BeginUpdate();
spStatusBar1->GetVisualAppearance()->Add(4,"c:\\exontrol\\images\\border.ebn");
spStatusBar1->GetVisualAppearance()->Add(5,"CP:4 2 2 -2 -2");
spStatusBar1->PutBackColorPanels(83886080);
spStatusBar1->PutDebug(VARIANT_TRUE);
spStatusBar1->PutFormat(L"(0/1:32),2,(3/4/5)");
spStatusBar1->EndUpdate();
```

The following **C#** sample shows "How do I draw a border for all panels":

```
axStatusBar1.BeginUpdate();
axStatusBar1.VisualAppearance.Add(4,"c:\\exontrol\\images\\border.ebn");
axStatusBar1.VisualAppearance.Add(5,"CP:4 2 2 -2 -2");
(axStatusBar1.GetOcx() as EXSTATUSBARLib.StatusBar).BackColorPanels = 0x5000000;
axStatusBar1.Debug = true;
axStatusBar1.Format = "(0/1:32),2,(3/4/5)";
axStatusBar1.EndUpdate();
```

The following **VFP** sample shows "How do I draw a border for all panels":

```
with thisform.StatusBar1
   .BeginUpdate
   .VisualAppearance.Add(4,"c:\exontrol\images\border.ebn")
   .VisualAppearance.Add(5,"CP:4 2 2 -2 -2")
```

```
    .BackColorPanels = 83886080
    .Debug = .T.
    .Format = "(0/1:32),2,(3/4/5)"
    .EndUpdate
endwith
```

property StatusBar.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. Use the Refresh method to refresh the control.

The following **VB** sample shows "Can I change the default border of the tooltip, using your EBN files":

```
With StatusBar1
   .BeginUpdate
   .ToolTipDelay = 1
   .ToolTipWidth = 364
   .VisualAppearance.Add 1,"c:\exontrol\images\normal.ebn"
   .Background(exToolTipAppearance) = 16777216
   .Format = "1"
   With .Panel(1)
      .Text = "This is a bit of text that should occurs when the cursor hovers the panel"
```

```
        .ToolTipText = .Text
        .ToolTipTitle = "Title"
        .Alignment = exAlignMiddleLeft
    End With
    .EndUpdate
End With
```

The following **VB.NET** sample shows "Can I change the default border of the tooltip, using your EBN files":

```
With AxStatusBar1
    .BeginUpdate
    .ToolTipDelay = 1
    .ToolTipWidth = 364
    .VisualAppearance.Add 1,"c:\exontrol\images\normal.ebn"

.set_Background(EXSTATUSBARLib.BackgroundPartEnum.exToolTipAppearance,16777216)
    .Format = "1"
    With .get_Panel(1)
        .Text = "This is a bit of text that should occurs when the cursor hovers the panel"
        .ToolTipText = .Text
        .ToolTipTitle = "Title"
        .Alignment = EXSTATUSBARLib.TextAlignEnum.exAlignMiddleLeft
    End With
    .EndUpdate
End With
```

The following **C++** sample shows "Can I change the default border of the tooltip, using your EBN files":

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

    #import "D:\\Exontrol\\ExStatusBar\\project\\Debug\\ExStatusBar.dll"
    using namespace EXSTATUSBARLib;
*/
```

```
EXSTATUSBARLib::IStatusBarPtr spStatusBar1 = GetDlgItem(IDC_STATUSBAR1)-
>GetControlUnknown();
spStatusBar1->BeginUpdate();
spStatusBar1->PutToolTipDelay(1);
spStatusBar1->PutToolTipWidth(364);
spStatusBar1->GetVisualAppearance()->Add(1,"c:\\exontrol\\images\\normal.ebn");
spStatusBar1->PutBackground(EXSTATUSBARLib::exToolTipAppearance,16777216);
spStatusBar1->PutFormat(L"1");
EXSTATUSBARLib::IPanelPtr var_Panel = spStatusBar1->GetPanel(long(1));
   var_Panel->PutText(L"This is a bit of text that should occurs when the cursor hovers the
panel");
   var_Panel->PutToolTipText(var_Panel->GetText());
   var_Panel->PutToolTipTitle(L"Title");
   var_Panel->PutAlignment(EXSTATUSBARLib::exAlignMiddleLeft);
spStatusBar1->EndUpdate();
```

The following **C#** sample shows "Can I change the default border of the tooltip, using your EBN files":

```
axStatusBar1.BeginUpdate();
axStatusBar1.ToolTipDelay = 1;
axStatusBar1.ToolTipWidth = 364;
axStatusBar1.VisualAppearance.Add(1,"c:\\exontrol\\images\\normal.ebn");
axStatusBar1.set_Background(EXSTATUSBARLib.BackgroundPartEnum.exToolTipAppearance,

axStatusBar1.Format = "1";
EXSTATUSBARLib.Panel var_Panel = axStatusBar1.get_Panel(1);
   var_Panel.Text = "This is a bit of text that should occurs when the cursor hovers the
panel";
   var_Panel.ToolTipText = var_Panel.Text;
   var_Panel.ToolTipTitle = "Title";
   var_Panel.Alignment = EXSTATUSBARLib.TextAlignEnum.exAlignMiddleLeft;
axStatusBar1.EndUpdate();
```

The following **VFP** sample shows "Can I change the default border of the tooltip, using your EBN files":

```
with thisform.StatusBar1
```

```
    .BeginUpdate
    .ToolTipDelay = 1
    .ToolTipWidth = 364
    .VisualAppearance.Add(1,"c:\exontrol\images\normal.ebn")
    .Background(64) = 16777216
    .Format = "1"
    with .Panel(1)
        .Text = "This is a bit of text that should occurs when the cursor hovers the panel"
        .ToolTipText = .Text
        .ToolTipTitle = "Title"
        .Alignment = 16
    endwith
    .EndUpdate
endwith
```

method StatusBar.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.

Type	Description

This method prevents the control from painting until the EndUpdate method is called. The BeginUpdate and EndUpdate methods increases the speed of loading your items, by preventing painting the control when it suffers any change. Once that BeginUpdate method was called, you have to make sure that EndUpdate method will be called too.

property StatusBar.BorderHeight as Long

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

Type	Description
Long	A long expression that specifies the height, in pixels of the border where the panels are displayed.

By default, the BorderHeight property is 0. Use the BorderWidth and BorderHeight properties to control the area where the panels are displayed. Use the Offset property to specify the padding to display the caption inside the panel. Use the OffsetPercent property to specify the padding to display the progress-bar inside the panel.

The following **VB** sample shows "Is there any option to increase the empty space on borders of the control":

```
With StatusBar1
   .BeginUpdate
   .BorderWidth = 20
   .BorderHeight = 20
   .VisualAppearance.Add 4,"c:\exontrol\images\border.ebn"
   .VisualAppearance.Add 5,"CP:4 1 1 -1 -1"
   .BackColorPanels = 83886080
   .Format = "1,2,3,4,(5/6/7/8)"
   .Debug = True
   .EndUpdate
End With
```

The following **VB.NET** sample shows "Is there any option to increase the empty space on borders of the control":

```
With AxStatusBar1
   .BeginUpdate
   .BorderWidth = 20
   .BorderHeight = 20
   .VisualAppearance.Add 4,"c:\exontrol\images\border.ebn"
   .VisualAppearance.Add 5,"CP:4 1 1 -1 -1"
   .GetOcx().BackColorPanels = &H5000000
   .Format = "1,2,3,4,(5/6/7/8)"
   .Debug = True
```

```
    .EndUpdate
End With
```

The following **C++** sample shows "Is there any option to increase the empty space on borders of the control":

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

    #import "D:\\Exontrol\\ExStatusBar\\project\\Debug\\ExStatusBar.dll"
    using namespace EXSTATUSBARLib;
*/
EXSTATUSBARLib::IStatusBarPtr spStatusBar1 = GetDlgItem(IDC_STATUSBAR1)-
>GetControlUnknown();
spStatusBar1->BeginUpdate();
spStatusBar1->PutBorderWidth(20);
spStatusBar1->PutBorderHeight(20);
spStatusBar1->GetVisualAppearance()->Add(4,"c:\\exontrol\\images\\border.ebn");
spStatusBar1->GetVisualAppearance()->Add(5,"CP:4 1 1 -1 -1");
spStatusBar1->PutBackColorPanels(83886080);
spStatusBar1->PutFormat(L"1,2,3,4,(5/6/7/8)");
spStatusBar1->PutDebug(VARIANT_TRUE);
spStatusBar1->EndUpdate();
```

The following **C#** sample shows "Is there any option to increase the empty space on borders of the control":

```
axStatusBar1.BeginUpdate();
axStatusBar1.BorderWidth = 20;
axStatusBar1.BorderHeight = 20;
axStatusBar1.VisualAppearance.Add(4,"c:\\exontrol\\images\\border.ebn");
axStatusBar1.VisualAppearance.Add(5,"CP:4 1 1 -1 -1");
(axStatusBar1.GetOcx() as EXSTATUSBARLib.StatusBar).BackColorPanels = 0x5000000;
axStatusBar1.Format = "1,2,3,4,(5/6/7/8)";
axStatusBar1.Debug = true;
axStatusBar1.EndUpdate();
```

The following **VFP** sample shows "Is there any option to increase the empty space on borders of the control":

```
with thisform.StatusBar1
   .BeginUpdate
   .BorderWidth = 20
   .BorderHeight = 20
   .VisualAppearance.Add(4,"c:\exontrol\images\border.ebn")
   .VisualAppearance.Add(5,"CP:4 1 1 -1 -1")
   .BackColorPanels = 83886080
   .Format = "1,2,3,4,(5/6/7/8)"
   .Debug = .T.
   .EndUpdate
endwith
```

property StatusBar.BorderWidth as Long

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

Type	Description
Long	A long expression that specifies the width, in pixels of the border where the panels are displayed.

By default, the BorderWidth property is 0. Use the BorderWidth and BorderHeight properties to control the area where the panels are displayed. Use the Offset property to specify the padding to display the caption inside the panel. Use the OffsetPercent property to specify the padding to display the progress-bar inside the panel.

The following **VB** sample shows "Is there any option to increase the empty space on borders of the control":

```
With StatusBar1
   .BeginUpdate
   .BorderWidth = 20
   .BorderHeight = 20
   .VisualAppearance.Add 4,"c:\exontrol\images\border.ebn"
   .VisualAppearance.Add 5,"CP:4 1 1 -1 -1"
   .BackColorPanels = 83886080
   .Format = "1,2,3,4,(5/6/7/8)"
   .Debug = True
   .EndUpdate
End With
```

The following **VB.NET** sample shows "Is there any option to increase the empty space on borders of the control":

```
With AxStatusBar1
   .BeginUpdate
   .BorderWidth = 20
   .BorderHeight = 20
   .VisualAppearance.Add 4,"c:\exontrol\images\border.ebn"
   .VisualAppearance.Add 5,"CP:4 1 1 -1 -1"
   .GetOcx().BackColorPanels = &H5000000
   .Format = "1,2,3,4,(5/6/7/8)"
   .Debug = True
```

```
    .EndUpdate
End With
```

The following **C++** sample shows "Is there any option to increase the empty space on borders of the control":

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

    #import "D:\\Exontrol\\ExStatusBar\\project\\Debug\\ExStatusBar.dll"
    using namespace EXSTATUSBARLib;
*/
EXSTATUSBARLib::IStatusBarPtr spStatusBar1 = GetDlgItem(IDC_STATUSBAR1)-
>GetControlUnknown();
spStatusBar1->BeginUpdate();
spStatusBar1->PutBorderWidth(20);
spStatusBar1->PutBorderHeight(20);
spStatusBar1->GetVisualAppearance()->Add(4,"c:\\exontrol\\images\\border.ebn");
spStatusBar1->GetVisualAppearance()->Add(5,"CP:4 1 1 -1 -1");
spStatusBar1->PutBackColorPanels(83886080);
spStatusBar1->PutFormat(L"1,2,3,4,(5/6/7/8)");
spStatusBar1->PutDebug(VARIANT_TRUE);
spStatusBar1->EndUpdate();
```

The following **C#** sample shows "Is there any option to increase the empty space on borders of the control":

```
axStatusBar1.BeginUpdate();
axStatusBar1.BorderWidth = 20;
axStatusBar1.BorderHeight = 20;
axStatusBar1.VisualAppearance.Add(4,"c:\\exontrol\\images\\border.ebn");
axStatusBar1.VisualAppearance.Add(5,"CP:4 1 1 -1 -1");
(axStatusBar1.GetOcx() as EXSTATUSBARLib.StatusBar).BackColorPanels = 0x5000000;
axStatusBar1.Format = "1,2,3,4,(5/6/7/8)";
axStatusBar1.Debug = true;
axStatusBar1.EndUpdate();
```

The following **VFP** sample shows "Is there any option to increase the empty space on borders of the control":

```
with thisform.StatusBar1
   .BeginUpdate
   .BorderWidth = 20
   .BorderHeight = 20
   .VisualAppearance.Add(4,"c:\exontrol\images\border.ebn")
   .VisualAppearance.Add(5,"CP:4 1 1 -1 -1")
   .BackColorPanels = 83886080
   .Format = "1,2,3,4,(5/6/7/8)"
   .Debug = .T.
   .EndUpdate
endwith
```

method StatusBar.ClearPanels ()

Clears the collection of panels in the control.

Type	Description

Use the ClearPanels property to remove all panels in the status bar control. Use the Format property to add and arrange the panels in your status bar control. Use the Panel property to access a panel in the status bar. If no panel is no accessed, the control display the Format property as an HTML caption. Use the Debug property to display the identifiers of the panels in the status bar.

property StatusBar.Debug as Boolean

Specifies whether the control displays debug information such of indentifiers of the panels.

Type	Description
Boolean	A Boolean expression that specifies whether the control displays the identifier of the panels in the status bar.

By default, the Debug property is True. Use the Debug property to display the identifiers of the panels in your status bar control. Use the Format property to add and arrange the panels in the status bar control. Use the Panel property to access a panel in the status bar control. The Index property of the panel indicates the index of the panel.

The following screen show shows the identifiers of the panels:

The following screen shot shows the status bar without displaying its identifiers:

So, the status bar has the following panels: 3, 11, 21, 33 and 44, and the Format property is *"<a1>link"[a=17]:64,**11**:64,((24;**21**/"Arrange the panels as you want using CRD strings"[a=17][ww])/24;**3**),(**33**/**44**):48.* The numbers that appear in bold, are the identifiers being displayed in the status bar.

property StatusBar.Enabled as Boolean

Enables or disables the control.

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

Use the Enabled property to disable the control. Use the Enabled property to disable a specified panel. When the control is disabled, all panels looks grayed.

The following **VB** sample shows "How can I disable the control":

```
With StatusBar1
   .BeginUpdate
   .Enabled = False
   .Images
"gBJJgBAIDAAGAAEAAQhYAf8Pf4hh0QihCJo2AEZjQAjEZFEaIEaEEaAIAkcbk0olUrlktl0vmExn
 & _
"/oFBoVDolFo1HpFJpVLplNp1PqFRqVTqlVq1XrFZrVbrldr1fsFhsVjslls1ntFptVrtltt1vuFxuVzul
 & _
"/wGBwWDwmFw2HxGJxWLxmNx0xiFdyOTh8Tf9ZymXx+QytcyNgz8r0OblWjyWds+m0ka1
 & _
"x3GO4NV3WeyvD2XJ5XL5nN51aiw+lfSj0gkUkAEllHanHI5j/cHg8EZf7w8vl8j4f/qfEZeB09/v
 & _
"NAOAEAwCjMBwFAEDwJBMDwLBYAP2/8Hv8/gAGAD8LQs9w/nhDY/oygIA="
   .VisualAppearance.Add 4,"c:\exontrol\images\border.ebn"
   .VisualAppearance.Add 5,"CP:4 1 1 -1 -1"
   .BackColorPanels = 83886080
   .Format = "1,2,3,4,(5/6/7/8)"
   .Panel(1).Image = 1
   .Panel(2).Text = "<img>1:4</img><img>1:4</img><img>1:4</img><img>1</img>
icons"
   With .Panel(3)
      .Text = "<img>2</img>"
      .Alignment = exAlignMiddleRight
   End With
   .EndUpdate
End With
```

The following **VB.NET** sample shows "How can I disable the control":

```
With AxStatusBar1
   .BeginUpdate
   .Enabled = False
   .Images
"gBJJgBAIDAAGAAEAAQhYAf8Pf4hh0QihCJo2AEZjQAjEZFEaIEaEEaAIAkcbk0oIUrlktl0vmExn
 & _
"/oFBoVDoIFo1HpFJpVLpINp1PqFRqVTqIVq1XrFZrVbrldr1fsFhsVjslls1ntFptVrtltt1vuFxuVzul
 & _
"/wGBwWDwmFw2HxGJxWLxmNx0xiFdyOTh8Tf9ZymXx+QytcyNgz8r0OblWjyWds+m0ka1
 & _
"x3GO4NV3WeyvD2XJ5XL5nN51aiw+lfSj0gkUkAEllHanHI5j/cHg8EZf7w8vl8j4f/qfEZeB09/v
 & _
"NAOAEAwCjMBwFAEDwJBMDwLBYAP2/8Hv8/gAGAD8LQs9w/nhDY/oygIA="
   .VisualAppearance.Add 4,"c:\exontrol\images\border.ebn"
   .VisualAppearance.Add 5,"CP:4 1 1 -1 -1"
   .GetOcx().BackColorPanels = &H5000000
   .Format = "1,2,3,4,(5/6/7/8)"
   .get_Panel(1).Image = 1
   .get_Panel(2).Text = "<img>1:4</img><img>1:4</img><img>1:4</img>
<img>1</img> icons"
   With .get_Panel(3)
      .Text = "<img>2</img>"
      .Alignment = EXSTATUSBARLib.TextAlignEnum.exAlignMiddleRight
   End With
   .EndUpdate
End With
```

The following **C++** sample shows "How can I disable the control":

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

   #import "C:\\WINNT\\system32\\ExStatusBar.dll"
   using namespace EXSTATUSBARLib;
```

```
*/
EXSTATUSBARLib::IStatusBarPtr spStatusBar1 = GetDlgItem(IDC_STATUSBAR1)-
>GetControlUnknown();
spStatusBar1->BeginUpdate();
spStatusBar1->PutEnabled(VARIANT_FALSE);
spStatusBar1-
>Images(_bstr_t("gBJJgBAIDAAGAAEAAQhYAf8Pf4hh0QihCJo2AEZjQAjEZFEaIEaEEaAIAkcbk
 +
"/oFBoVDoIFo1HpFJpVLpINp1PqFRqVTqIVq1XrFZrVbrIdr1fsFhsVjslls1ntFptVrtltt1vuFxuVzul
 +
"/wGBwWDwmFw2HxGJxWLxmNx0xiFdyOTh8Tf9ZymXx+QytcyNgz8r0OblWjyWds+m0ka1
 +
"x3GO4NV3WeyvD2XJ5XL5nN51aiw+lfSj0gkUkAEllHanHI5j/cHg8EZf7w8vl8j4f/qfEZeB09/v
 +
"NAOAEAwCjMBwFAEDwJBMDwLBYAP2/8Hv8/gAGAD8LQs9w/nhDY/oygIA=");
spStatusBar1->GetVisualAppearance()->Add(4,"c:\\exontrol\\images\\border.ebn");
spStatusBar1->GetVisualAppearance()->Add(5,"CP:4 1 1 -1 -1");
spStatusBar1->PutBackColorPanels(83886080);
spStatusBar1->PutFormat(L"1,2,3,4,(5/6/7/8)");
spStatusBar1->GetPanel(long(1))->PutImage(1);
spStatusBar1->GetPanel(long(2))->PutText(L"<img>1:4</img><img>1:4</img>
<img>1:4</img><img>1</img> icons");
EXSTATUSBARLib::IPanelPtr var_Panel = spStatusBar1->GetPanel(long(3));
   var_Panel->PutText(L"<img>2</img>");
   var_Panel->PutAlignment(EXSTATUSBARLib::exAlignMiddleRight);
spStatusBar1->EndUpdate();
```

The following **C#** sample shows "How can I disable the control":

```
axStatusBar1.BeginUpdate();
axStatusBar1.Enabled = false;
axStatusBar1.Images("gBJJgBAIDAAGAAEAAQhYAf8Pf4hh0QihCJo2AEZjQAjEZFEaIEaEEaAIA
 +
"/oFBoVDoIFo1HpFJpVLpINp1PqFRqVTqIVq1XrFZrVbrIdr1fsFhsVjslls1ntFptVrtltt1vuFxuVzul
 +
"/wGBwWDwmFw2HxGJxWLxmNx0xiFdyOTh8Tf9ZymXx+QytcyNgz8r0OblWjyWds+m0ka1
 +
```

```
"x3GO4NV3WeyvD2XJ5XL5nN51aiw+lfSj0gkUkAEllHanHI5j/cHg8EZf7w8vl8j4f/qfEZeB09/v
 +
"NAOAEAwCjMBwFAEDwJBMDwLBYAP2/8Hv8/gAGAD8LQs9w/nhDY/oygIA=");
axStatusBar1.VisualAppearance.Add(4,"c:\\exontrol\\images\\border.ebn");
axStatusBar1.VisualAppearance.Add(5,"CP:4 1 1 -1 -1");
(axStatusBar1.GetOcx() as EXSTATUSBARLib.StatusBar).BackColorPanels = 0x5000000;
axStatusBar1.Format = "1,2,3,4,(5/6/7/8)";
axStatusBar1.get_Panel(1).Image = 1;
axStatusBar1.get_Panel(2).Text = "<img>1:4</img><img>1:4</img><img>1:4</img>
<img>1</img> icons";
EXSTATUSBARLib.Panel var_Panel = axStatusBar1.get_Panel(3);
   var_Panel.Text = "<img>2</img>";
   var_Panel.Alignment = EXSTATUSBARLib.TextAlignEnum.exAlignMiddleRight;
axStatusBar1.EndUpdate();
```

The following **VFP** sample shows "How can I disable the control":

```
with thisform.StatusBar1
   .BeginUpdate
   .Enabled = .F.
   var_s =
"gBJJgBAIDAAGAAEAAQhYAf8Pf4hh0QihCJo2AEZjQAjEZFEaIEaEEaAIAkcbk0oIUrlktl0vmExn

   var_s = var_s +
"oFBoVDoIFo1HpFJpVLplNp1PqFRqVTqlVq1XrFZrVbrldr1fsFhsVjslls1ntFptVrtltt1vuFxuVzul1

   var_s = var_s +
"wGBwWDwmFw2HxGJxWLxmNx0xiFdyOTh8Tf9ZymXx+QytcyNgz8r0OblWjyWds+m0ka1V

   var_s = var_s +
"3GO4NV3WeyvD2XJ5XL5nN51aiw+lfSj0gkUkAEllHanHI5j/cHg8EZf7w8vl8j4f/qfEZeB09/vj

   var_s = var_s +
"AOAEAwCjMBwFAEDwJBMDwLBYAP2/8Hv8/gAGAD8LQs9w/nhDY/oygIA="
   .Images(var_s)
   .VisualAppearance.Add(4,"c:\exontrol\images\border.ebn")
   .VisualAppearance.Add(5,"CP:4 1 1 -1 -1")
```

```
    .BackColorPanels = 83886080
    .Format = "1,2,3,4,(5/6/7/8)"
    .Panel(1).Image = 1
    .Panel(2).Text = "<img>1:4</img><img>1:4</img><img>1:4</img><img>1</img>
icons"
    with .Panel(3)
        .Text = "<img>2</img>"
        .Alignment = 18
    endwith
    .EndUpdate
endwith
```

method StatusBar.EndUpdate ()

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

Type	Description

The BeginUpdate and EndUpdate methods increases the speed of loading your items, by preventing painting the control when it suffers any change. Once that BeginUpdate method was called, you have to make sure that EndUpdate method will be called too.

property StatusBar.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 StatusBar.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 String expression that indicates the result after executing the Template.

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

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

```
Debug.Print StatusBar1.ExecuteTemplate("Items.ItemBar(FirstVisibleItem(),``,1)")
```

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

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

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

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

"\n\r" (newline) characters.

An instruction can be one of the following:

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

The Template supports the following general functions:

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

property StatusBar.Font as IFontDisp

Retrieves or sets the control's font.

Type	Description
IFontDisp	A Font object that specifies the font to display the panels.

Use the Font property to specify the font to display the panels. Use the <font> HTM tag to apply a different font for portion of text in the caption of the panel. Use the Text property to specify the caption of the panel. Use the ToolTipFont property to specify the the font to display the tooltip when the cursor hovers a panel. Use the ForeColor property to specify the control's foreground color. Use the Bold property to show the panel's caption in bold. Use the Italic property to show the panel's caption in italic. Use the Underline property to underline the panel's caption. Use the StrikeOut property to show the panel's caption in strikeout.

The following **VB** sample shows "How can I change the control's font":

```
With StatusBar1
   .BeginUpdate
   .VisualAppearance.Add 4,"c:\exontrol\images\border.ebn"
   .VisualAppearance.Add 5,"CP:4 1 1 -1 -1"
   .BackColorPanels = 83886080
   Set f = CreateObject("StdFont")
   With f
      .Name = "Verdana"
      .Size = 12
   End With
   .Font = f
   .Format = """static text""[fg=255][a=17],11,22,(33/44)"
   .EndUpdate
End With
```

The following **VB.NET** sample shows "How can I change the control's font":

```
Dim f
With AxStatusBar1
   .BeginUpdate
   .VisualAppearance.Add 4,"c:\exontrol\images\border.ebn"
   .VisualAppearance.Add 5,"CP:4 1 1 -1 -1"
```

```
    .GetOcx().BackColorPanels = &H5000000
    f = CreateObject("StdFont")
    With f
        .Name = "Verdana"
        .Size = 12
    End With
    .Font = f
    .Format = """static text""[fg=255][a=17],11,22,(33/44)"
    .EndUpdate
End With
```

The following **C++** sample shows "How can I change the control's font":

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

    #import "D:\\Exontrol\\ExStatusBar\\project\\Debug\\ExStatusBar.dll"
    using namespace EXSTATUSBARLib;
*/
EXSTATUSBARLib::IStatusBarPtr spStatusBar1 = GetDlgItem(IDC_STATUSBAR1)-
>GetControlUnknown();
spStatusBar1->BeginUpdate();
spStatusBar1->GetVisualAppearance()->Add(4,"c:\\exontrol\\images\\border.ebn");
spStatusBar1->GetVisualAppearance()->Add(5,"CP:4 1 1 -1 -1");
spStatusBar1->PutBackColorPanels(83886080);
/*
    Includes the definition for CreateObject function like follows:

    #include <comdef.h>
    IUnknownPtr CreateObject( BSTR Object )
    {
        IUnknownPtr spResult;
        spResult.CreateInstance( Object );
        return spResult;
    };
```

```
*/
/*
   Copy and paste the following directives to your header file as
   it defines the namespace 'stdole' for the library: 'OLE Automation'

   #import "C:\\WINNT\\System32\\stdole2.tlb"
*/
stdole::FontPtr f = ::CreateObject(L"StdFont");
   f->PutName(L"Verdana");
   f->PutSize(_variant_t(long(12)));
spStatusBar1->PutFont(IFontDispPtr(((stdole::FontPtr)(f))));
spStatusBar1->PutFormat(L"\"static text\"[fg=255][a=17],11,22,(33/44)");
spStatusBar1->EndUpdate();
```

The following **C#** sample shows "How can I change the control's font":

```
axStatusBar1.BeginUpdate();
axStatusBar1.VisualAppearance.Add(4,"c:\\exontrol\\images\\border.ebn");
axStatusBar1.VisualAppearance.Add(5,"CP:4 1 1 -1 -1");
(axStatusBar1.GetOcx() as EXSTATUSBARLib.StatusBar).BackColorPanels = 0x5000000;
stdole.IFontDisp f = new stdole.StdFont() as stdole.IFontDisp;
   f.Name = "Verdana";
   f.Size = 12;
axStatusBar1.Font = (f as stdole.IFontDisp);
axStatusBar1.Format = "\"static text\"[fg=255][a=17],11,22,(33/44)";
axStatusBar1.EndUpdate();
```

The following **VFP** sample shows "How can I change the control's font":

```
with thisform.StatusBar1
   .BeginUpdate
   .VisualAppearance.Add(4,"c:\exontrol\images\border.ebn")
   .VisualAppearance.Add(5,"CP:4 1 1 -1 -1")
   .BackColorPanels = 83886080
   f = CreateObject("StdFont")
   with f
      .Name = "Verdana"
```

```
      .Size = 12
    endwith
    .Font = f
    .Format = ""+chr(34)+"static text"+chr(34)+"[fg=255][a=17],11,22,(33/44)"
    .EndUpdate
endwith
```

property StatusBar.ForeColor as Color

Specifies the control's foreground color.

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

Use the ForeColor property to specify the control's foreground color. Use the <fgcolor> HTML tag to specify a different foreground color for portion of text in the panel. Use the Text property to assign a caption to a panel. Use the BackColor property to specify the control's background color.

The following **VB** sample shows "How do I change the control's foreground color":

```
With StatusBar1
   .BeginUpdate
   .ForeColor = 7895160
   .VisualAppearance.Add 4,"c:\exontrol\images\border.ebn"
   .VisualAppearance.Add 5,"CP:4 1 1 -1 -1"
   .BackColorPanels = 83886080
   .Format = "1,2,3,4,(5/6/7/8)"
   .Debug = True
   .EndUpdate
End With
```

The following **VB.NET** sample shows "How do I change the control's foreground color":

```
With AxStatusBar1
   .BeginUpdate
   .ForeColor = Color.FromArgb(120,120,120)
   .VisualAppearance.Add 4,"c:\exontrol\images\border.ebn"
   .VisualAppearance.Add 5,"CP:4 1 1 -1 -1"
   .GetOcx().BackColorPanels = &H5000000
   .Format = "1,2,3,4,(5/6/7/8)"
   .Debug = True
   .EndUpdate
End With
```

The following **C++** sample shows "How do I change the control's foreground color":

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

   #import "D:\\Exontrol\\ExStatusBar\\project\\Debug\\ExStatusBar.dll"
   using namespace EXSTATUSBARLib;
*/
EXSTATUSBARLib::IStatusBarPtr spStatusBar1 = GetDlgItem(IDC_STATUSBAR1)-
>GetControlUnknown();
spStatusBar1->BeginUpdate();
spStatusBar1->PutForeColor(7895160);
spStatusBar1->GetVisualAppearance()->Add(4,"c:\\exontrol\\images\\border.ebn");
spStatusBar1->GetVisualAppearance()->Add(5,"CP:4 1 1 -1 -1");
spStatusBar1->PutBackColorPanels(83886080);
spStatusBar1->PutFormat(L"1,2,3,4,(5/6/7/8)");
spStatusBar1->PutDebug(VARIANT_TRUE);
spStatusBar1->EndUpdate();
```

The following **C#** sample shows "How do I change the control's foreground color":

```
axStatusBar1.BeginUpdate();
axStatusBar1.ForeColor = Color.FromArgb(120,120,120);
axStatusBar1.VisualAppearance.Add(4,"c:\\exontrol\\images\\border.ebn");
axStatusBar1.VisualAppearance.Add(5,"CP:4 1 1 -1 -1");
(axStatusBar1.GetOcx() as EXSTATUSBARLib.StatusBar).BackColorPanels = 0x5000000;
axStatusBar1.Format = "1,2,3,4,(5/6/7/8)";
axStatusBar1.Debug = true;
axStatusBar1.EndUpdate();
```

The following **VFP** sample shows "How do I change the control's foreground color":

```
with thisform.StatusBar1
   .BeginUpdate
   .ForeColor = 7895160
   .VisualAppearance.Add(4,"c:\exontrol\images\border.ebn")
   .VisualAppearance.Add(5,"CP:4 1 1 -1 -1")
   .BackColorPanels = 83886080
```

```
   .Format = "1,2,3,4,(5/6/7/8)"
   .Debug = .T.
   .EndUpdate
endwith
```

property StatusBar.Format as String

Specifies the CRD format to arrange the objects inside the control.

Type	Description
String	A CRD String expression that specifies the arrangement of the panels and captions in the status bar. **This is a CRD string.**

By default, the Format property is empty, so no panels are displayed. Use the Format property to add an arrange the panels in the status bar. Use the Panel property to access panels in the status bar. Use the Debug property to display the identifiers of the panels in the status bar. *Use the BackColorPanels property to specify the visual appearance of the borders around the panels.* For instance, if the Format property is "1,2" the status bar displays two panels with the identifiers 1 and 2.

The following **VB** sample shows how can I add two panels to the status bar control:

```
With StatusBar1
   .BeginUpdate
   .VisualAppearance.Add 4,"c:\exontrol\images\border.ebn"
   .VisualAppearance.Add 5,"CP:4 1 1 -1 -1"
   .BackColorPanels = 83886080
   .Format = "1,2"
   .Panel(1).Text = "Panel 1"
   .Panel(2).Text = "Panel 2"
   .EndUpdate
End With
```

The following **VB** sample shows how can I add three panels aligned from top to bottom:

```
With StatusBar1
   .BeginUpdate
   .BackColor = -2147483633
   .Images
"gBJJgBAIDAAGAAEAAQhYAf8Pf4hh0QihCJo2AEZjQAjEZFEaIEaEEaAIAkcbk0oIUrlktl0vmExn
 & _
"/oFBoVDoIFo1HpFJpVLpINp1PqFRqVTqIVq1XrFZrVbrIdr1fsFhsVjslls1ntFptVrtltt1vuFxuVzul
 & _
"/wGBwWDwmFw2HxGJxWLxmNx0xiFdyOTh8Tf9ZymXx+QytcyNgz8r0OblWjyWds+m0ka1
```

```
 & _
"x3GO4NV3WeyvD2XJ5XL5nN51aiw+lfSj0gkUkAEllHanHI5j/cHg8EZf7w8vl8j4f/qfEZeB09/v
 & _
"NAOAEAwCjMBwFAEDwJBMDwLBYAP2/8Hv8/gAGAD8LQs9w/nhDY/oygIA="
   .VisualAppearance.Add 4,"c:\exontrol\images\border.ebn"
   .VisualAppearance.Add 5,"CP:4 1 1 -1 -1"
   .BackColorPanels = 83886080
   .Format = "1/2/3"
   .Panel(1).Text = "Panel 1"
   .Panel(2).Text = "Panel 2"
   .Panel(3).Text = "Panel 3"
   .EndUpdate
End With
```

The following **VB.NET** sample shows how can I add two panels to the status bar control:

```
With AxStatusBar1
   .BeginUpdate
   .VisualAppearance.Add 4,"c:\exontrol\images\border.ebn"
   .VisualAppearance.Add 5,"CP:4 1 1 -1 -1"
   .GetOcx().BackColorPanels = &H5000000
   .Format = "1,2"
   .get_Panel(1).Text = "Panel 1"
   .get_Panel(2).Text = "Panel 2"
   .EndUpdate
End With
```

The following **VB.NET** sample shows how can I add three panels aligned from top to bottom:

```
With AxStatusBar1
   .BeginUpdate
   .GetOcx().BackColor = &H8000000f
   .Images
"gBJJgBAIDAAGAAEAAQhYAf8Pf4hh0QihCJo2AEZjQAjEZFEaIEaEEaAIAkcbk0oIUrlktl0vmExn
 & _
"/oFBoVDoIFo1HpFJpVLplNp1PqFRqVTqlVq1XrFZrVbrldr1fsFhsVjslls1ntFptVrtltt1vuFxuVzul
 & _
```

```
"/wGBwWDwmFw2HxGJxWLxmNx0xiFdyOTh8Tf9ZymXx+QytcyNgz8r0OblWjyWds+m0ka1
 & _
"x3GO4NV3WeyvD2XJ5XL5nN51aiw+lfSj0gkUkAEllHanHI5j/cHg8EZf7w8vl8j4f/qfEZeB09/v
 & _
"NAOAEAwCjMBwFAEDwJBMDwLBYAP2/8Hv8/gAGAD8LQs9w/nhDY/oygIA="
   .VisualAppearance.Add 4,"c:\exontrol\images\border.ebn"
   .VisualAppearance.Add 5,"CP:4 1 1 -1 -1"
   .GetOcx().BackColorPanels = &H5000000
   .Format = "1/2/3"
   .get_Panel(1).Text = "Panel 1"
   .get_Panel(2).Text = "Panel 2"
   .get_Panel(3).Text = "Panel 3"
   .EndUpdate
End With
```

The following **C++** sample shows how can I add two panels to the status bar control:

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

   #import "D:\\Exontrol\\ExStatusBar\\project\\Debug\\ExStatusBar.dll"
   using namespace EXSTATUSBARLib;
*/
EXSTATUSBARLib::IStatusBarPtr spStatusBar1 = GetDlgItem(IDC_STATUSBAR1)-
>GetControlUnknown();
spStatusBar1->BeginUpdate();
spStatusBar1->GetVisualAppearance()->Add(4,"c:\\exontrol\\images\\border.ebn");
spStatusBar1->GetVisualAppearance()->Add(5,"CP:4 1 1 -1 -1");
spStatusBar1->PutBackColorPanels(83886080);
spStatusBar1->PutFormat(L"1,2");
spStatusBar1->GetPanel(long(1))->PutText(L"Panel 1");
spStatusBar1->GetPanel(long(2))->PutText(L"Panel 2");
spStatusBar1->EndUpdate();
```

The following **C++** sample shows how can I add three panels aligned from top to bottom:

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

   #import "D:\\Exontrol\\ExStatusBar\\project\\Debug\\ExStatusBar.dll"
   using namespace EXSTATUSBARLib;
*/
EXSTATUSBARLib::IStatusBarPtr spStatusBar1 = GetDlgItem(IDC_STATUSBAR1)-
>GetControlUnknown();
spStatusBar1->BeginUpdate();
spStatusBar1->PutBackColor(-2147483633);
spStatusBar1-
>Images(_bstr_t("gBJJgBAIDAAGAAEAAQhYAf8Pf4hh0QihCJo2AEZjQAjEZFEaIEaEEaAIAkcbk
 +
"/oFBoVDolFo1HpFJpVLplNp1PqFRqVTqlVq1XrFZrVbrldr1fsFhsVjslls1ntFptVrtltt1vuFxuVzul
 +
"/wGBwWDwmFw2HxGJxWLxmNx0xiFdyOTh8Tf9ZymXx+QytcyNgz8r0OblWjyWds+m0ka1
 +
"x3GO4NV3WeyvD2XJ5XL5nN51aiw+lfSj0gkUkAEllHanHI5j/cHg8EZf7w8vl8j4f/qfEZeB09/v
 +
"NAOAEAwCjMBwFAEDwJBMDwLBYAP2/8Hv8/gAGAD8LQs9w/nhDY/oygIA=");
spStatusBar1->GetVisualAppearance()->Add(4,"c:\\exontrol\\images\\border.ebn");
spStatusBar1->GetVisualAppearance()->Add(5,"CP:4 1 1 -1 -1");
spStatusBar1->PutBackColorPanels(83886080);
spStatusBar1->PutFormat(L"1/2/3");
spStatusBar1->GetPanel(long(1))->PutText(L"Panel 1");
spStatusBar1->GetPanel(long(2))->PutText(L"Panel 2");
spStatusBar1->GetPanel(long(3))->PutText(L"Panel 3");
spStatusBar1->EndUpdate();
```

The following **C#** sample shows how can I add two panels to the status bar control:

```
axStatusBar1.BeginUpdate();
axStatusBar1.VisualAppearance.Add(4,"c:\\exontrol\\images\\border.ebn");
axStatusBar1.VisualAppearance.Add(5,"CP:4 1 1 -1 -1");
(axStatusBar1.GetOcx() as EXSTATUSBARLib.StatusBar).BackColorPanels = 0x5000000;
```

```
axStatusBar1.Format = "1,2";
axStatusBar1.get_Panel(1).Text = "Panel 1";
axStatusBar1.get_Panel(2).Text = "Panel 2";
axStatusBar1.EndUpdate();
```

The following **C#** sample shows how can I add three panels aligned from top to bottom:

```
axStatusBar1.BeginUpdate();
(axStatusBar1.GetOcx() as EXSTATUSBARLib.StatusBar).BackColor = 0x8000000f;
axStatusBar1.Images("gBJJgBAIDAAGAAEAAQhYAf8Pf4hh0QihCJo2AEZjQAjEZFEaIEaEEaAIA
 +
"/oFBoVDoIFo1HpFJpVLpINp1PqFRqVTqIVq1XrFZrVbrIdr1fsFhsVjslls1ntFptVrtltt1vuFxuVzul
 +
"/wGBwWDwmFw2HxGJxWLxmNx0xiFdyOTh8Tf9ZymXx+QytcyNgz8r0OblWjyWds+m0ka1
 +
"x3GO4NV3WeyvD2XJ5XL5nN51aiw+lfSj0gkUkAEllHanHI5j/cHg8EZf7w8vl8j4f/qfEZeB09/v
 +
"NAOAEAwCjMBwFAEDwJBMDwLBYAP2/8Hv8/gAGAD8LQs9w/nhDY/oygIA=");
axStatusBar1.VisualAppearance.Add(4,"c:\\exontrol\\images\\border.ebn");
axStatusBar1.VisualAppearance.Add(5,"CP:4 1 1 -1 -1");
(axStatusBar1.GetOcx() as EXSTATUSBARLib.StatusBar).BackColorPanels = 0x5000000;
axStatusBar1.Format = "1/2/3";
axStatusBar1.get_Panel(1).Text = "Panel 1";
axStatusBar1.get_Panel(2).Text = "Panel 2";
axStatusBar1.get_Panel(3).Text = "Panel 3";
axStatusBar1.EndUpdate();
```

The following **VFP** sample shows how can I add two panels to the status bar control:

```
with thisform.StatusBar1
   .BeginUpdate
   .VisualAppearance.Add(4,"c:\exontrol\images\border.ebn")
   .VisualAppearance.Add(5,"CP:4 1 1 -1 -1")
   .BackColorPanels = 83886080
   .Format = "1,2"
   .Panel(1).Text = "Panel 1"
   .Panel(2).Text = "Panel 2"
   .EndUpdate
```

```
endwith
```

The following **VFP** sample shows how can I add three panels aligned from top to bottom:

```
with thisform.StatusBar1
   .BeginUpdate
   .BackColor = -2147483633
   var_s =
"gBJJgBAIDAAGAAEAAQhYAf8Pf4hh0QihCJo2AEZjQAjEZFEaIEaEEaAIAkcbk0oIUrlktl0vmExn

   var_s = var_s +
"oFBoVDoIFo1HpFJpVLplNp1PqFRqVTqlVq1XrFZrVbrldr1fsFhsVjslls1ntFptVrtltt1vuFxuVzul1

   var_s = var_s +
"wGBwWDwmFw2HxGJxWLxmNx0xiFdyOTh8Tf9ZymXx+QytcyNgz8r0OblWjyWds+m0ka1V

   var_s = var_s +
"3GO4NV3WeyvD2XJ5XL5nN51aiw+lfSj0gkUkAEllHanHI5j/cHg8EZf7w8vl8j4f/qfEZeB09/vj

   var_s = var_s +
"AOAEAwCjMBwFAEDwJBMDwLBYAP2/8Hv8/gAGAD8LQs9w/nhDY/oygIA="
   .Images(var_s)
   .VisualAppearance.Add(4,"c:\exontrol\images\border.ebn")
   .VisualAppearance.Add(5,"CP:4 1 1 -1 -1")
   .BackColorPanels = 83886080
   .Format = "1/2/3"
   .Panel(1).Text = "Panel 1"
   .Panel(2).Text = "Panel 2"
   .Panel(3).Text = "Panel 3"
   .EndUpdate
endwith
```

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

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.

The following **VB** sample shows "Can I change the visual effect, appearance for the anchor, hyperlink elements, in HTML captions":

```
With StatusBar1
   .BeginUpdate
   .FormatAnchor(True) = "<b><u><fgcolor=FF0000> </fgcolor></u></b>"
   .VisualAppearance.Add 4,"c:\exontrol\images\border.ebn"
   .VisualAppearance.Add 5,"CP:4 1 1 -1 -1"
   .BackColorPanels = 83886080
   .Format = "1,2,3"
   .Panel(1).Text = "<a1>link 1</a>"
   .Panel(2).Text = "<a2>link 2</a>"
   .Panel(3).Text = "<a3>link 3</a>"
   .EndUpdate
End With
```

The following **VB.NET** sample shows "Can I change the visual effect, appearance for the anchor, hyperlink elements, in HTML captions":

```
With AxStatusBar1
   .BeginUpdate
   .set_FormatAnchor(True,"<b><u><fgcolor=FF0000> </fgcolor></u></b>")
   .VisualAppearance.Add 4,"c:\exontrol\images\border.ebn"
   .VisualAppearance.Add 5,"CP:4 1 1 -1 -1"
   .GetOcx().BackColorPanels = &H5000000
   .Format = "1,2,3"
   .get_Panel(1).Text = "<a1>link 1</a>"
   .get_Panel(2).Text = "<a2>link 2</a>"
   .get_Panel(3).Text = "<a3>link 3</a>"
   .EndUpdate
End With
```

The following **C++** sample shows "Can I change the visual effect, appearance for the anchor, hyperlink elements, in HTML captions":

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

   #import "D:\\Exontrol\\ExStatusBar\\project\\Debug\\ExStatusBar.dll"
```

```
   using namespace EXSTATUSBARLib;
*/
EXSTATUSBARLib::IStatusBarPtr spStatusBar1 = GetDlgItem(IDC_STATUSBAR1)-
>GetControlUnknown();
spStatusBar1->BeginUpdate();
spStatusBar1->PutFormatAnchor(VARIANT_TRUE,L"<b><u><fgcolor=FF0000>
</fgcolor></u></b>");
spStatusBar1->GetVisualAppearance()->Add(4,"c:\\exontrol\\images\\border.ebn");
spStatusBar1->GetVisualAppearance()->Add(5,"CP:4 1 1 -1 -1");
spStatusBar1->PutBackColorPanels(83886080);
spStatusBar1->PutFormat(L"1,2,3");
spStatusBar1->GetPanel(long(1))->PutText(L"<a1>link 1</a>");
spStatusBar1->GetPanel(long(2))->PutText(L"<a2>link 2</a>");
spStatusBar1->GetPanel(long(3))->PutText(L"<a3>link 3</a>");
spStatusBar1->EndUpdate();
```

The following **C#** sample shows "Can I change the visual effect, appearance for the anchor, hyperlink elements, in HTML captions":

```
axStatusBar1.BeginUpdate();
axStatusBar1.set_FormatAnchor(true,"<b><u><fgcolor=FF0000> </fgcolor></u>
</b>");
axStatusBar1.VisualAppearance.Add(4,"c:\\exontrol\\images\\border.ebn");
axStatusBar1.VisualAppearance.Add(5,"CP:4 1 1 -1 -1");
(axStatusBar1.GetOcx() as EXSTATUSBARLib.StatusBar).BackColorPanels = 0x5000000;
axStatusBar1.Format = "1,2,3";
axStatusBar1.get_Panel(1).Text = "<a1>link 1</a>";
axStatusBar1.get_Panel(2).Text = "<a2>link 2</a>";
axStatusBar1.get_Panel(3).Text = "<a3>link 3</a>";
axStatusBar1.EndUpdate();
```

The following **VFP** sample shows "Can I change the visual effect, appearance for the anchor, hyperlink elements, in HTML captions":

```
with thisform.StatusBar1
   .BeginUpdate
   .FormatAnchor(.T.) = "<b><u><fgcolor=FF0000> </fgcolor></u></b>"
   .VisualAppearance.Add(4,"c:\exontrol\images\border.ebn")
```

```
    .VisualAppearance.Add(5,"CP:4 1 1 -1 -1")
    .BackColorPanels = 83886080
    .Format = "1,2,3"
    .Panel(1).Text = "<a1>link 1</a>"
    .Panel(2).Text = "<a2>link 2</a>"
    .Panel(3).Text = "<a3>link 3</a>"
    .EndUpdate
endwith
```

property StatusBar.HTMLPicture(Key as String) as Variant

Adds or replaces a picture in HTML captions.

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

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

The following **VB** sample shows "Can I displays a custom size picture to panels":

```
With StatusBar1
   .BeginUpdate
   .VisualAppearance.Add 4,"c:\exontrol\images\border.ebn"
   .VisualAppearance.Add 5,"CP:4 1 1 -1 -1"
   .BackColorPanels = 83886080
   .HTMLPicture("pic1") = "c:\exontrol\images\zipdisk.gif"
```

```
    .Format = "1,2,3,4"
    .Panel(1).Text = "<img>pic1</img>"
    .EndUpdate
End With
```

The following **VB.NET** sample shows "Can I displays a custom size picture to panels":

```
With AxStatusBar1
    .BeginUpdate
    .VisualAppearance.Add 4,"c:\exontrol\images\border.ebn"
    .VisualAppearance.Add 5,"CP:4 1 1 -1 -1"
    .GetOcx().BackColorPanels = &H5000000
    .set_HTMLPicture("pic1","c:\exontrol\images\zipdisk.gif")
    .Format = "1,2,3,4"
    .get_Panel(1).Text = "<img>pic1</img>"
    .EndUpdate
End With
```

The following **C++** sample shows "Can I displays a custom size picture to panels":

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

    #import "D:\\Exontrol\\ExStatusBar\\project\\Debug\\ExStatusBar.dll"
    using namespace EXSTATUSBARLib;
*/
EXSTATUSBARLib::IStatusBarPtr spStatusBar1 = GetDlgItem(IDC_STATUSBAR1)-
>GetControlUnknown();
spStatusBar1->BeginUpdate();
spStatusBar1->GetVisualAppearance()->Add(4,"c:\\exontrol\\images\\border.ebn");
spStatusBar1->GetVisualAppearance()->Add(5,"CP:4 1 1 -1 -1");
spStatusBar1->PutBackColorPanels(83886080);
spStatusBar1->PutHTMLPicture(L"pic1","c:\\exontrol\\images\\zipdisk.gif");
spStatusBar1->PutFormat(L"1,2,3,4");
spStatusBar1->GetPanel(long(1))->PutText(L"<img>pic1</img>");
spStatusBar1->EndUpdate();
```

The following **C#** sample shows "Can I displays a custom size picture to panels":

```
axStatusBar1.BeginUpdate();
axStatusBar1.VisualAppearance.Add(4,"c:\\exontrol\\images\\border.ebn");
axStatusBar1.VisualAppearance.Add(5,"CP:4 1 1 -1 -1");
(axStatusBar1.GetOcx() as EXSTATUSBARLib.StatusBar).BackColorPanels = 0x5000000;
axStatusBar1.set_HTMLPicture("pic1","c:\\exontrol\\images\\zipdisk.gif");
axStatusBar1.Format = "1,2,3,4";
axStatusBar1.get_Panel(1).Text = "<img>pic1</img>";
axStatusBar1.EndUpdate();
```

The following **VFP** sample shows "Can I displays a custom size picture to panels":

```
with thisform.StatusBar1
   .BeginUpdate
   .VisualAppearance.Add(4,"c:\exontrol\images\border.ebn")
   .VisualAppearance.Add(5,"CP:4 1 1 -1 -1")
   .BackColorPanels = 83886080
   .HTMLPicture("pic1") = "c:\exontrol\images\zipdisk.gif"
   .Format = "1,2,3,4"
   .Panel(1).Text = "<img>pic1</img>"
   .EndUpdate
endwith
```

property StatusBar.hWnd as Long

Retrieves the control's window handle.

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

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

method StatusBar.Images (Handle as Variant)

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

Type	Description
Handle as Variant	The Handle parameter can be: • A string expression that specifies the ICO file to add. The ICO file format is an image file format for computer icons in Microsoft Windows. ICO files contain one or more small images at multiple sizes and color depths, such that they may be scaled appropriately. For instance, Images("c:\temp\copy.ico") method adds the sync.ico file to the control's Images collection *(string, loads the icon using its path)* • A string expression that indicates the BASE64 encoded string that holds the icons list. Use the Exontrol's ExImages tool to save/load your icons as BASE64 encoded format. In this case the string may begin with "gBJJ..." *(string, loads icons using base64 encoded string)* • A reference to a Microsoft ImageList control (mscomctl.ocx, MSComctlLib.ImageList type) that holds the icons to add *(object, loads icons from a Microsoft ImageList control)* • A reference to a Picture (IPictureDisp implementation) that holds the icon to add. For instance, the VB's LoadPicture (Function LoadPicture([FileName], [Size], [ColorDepth], [X], [Y]) As IPictureDisp) or LoadResPicture (Function LoadResPicture(id, restype As Integer) As IPictureDisp) returns a picture object (object, loads icon from a Picture object) • A long expression that identifies a handle to an Image List Control (the Handle should be of HIMAGELIST type). On 64-bit platforms, the Handle parameter must be a Variant of LongLong / LONG_PTR data type (signed 64-bit (8-byte) integers), saved under llVal field, as VT_I8 type. The LONGLONG / LONG_PTR is __int64, a 64-bit integer. For instance, in C++ you can use as Images(COleVariant((LONG_PTR)hImageList)) or Images(COleVariant(

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

The user can add images at design time, by drag and drop files to combo's image holder. The ImageSize property defines the size (width/height) of the icons within the control's Images collection. Use the RepaceIcon method to add, remove or clear icons in the control's images collection. Use the Image property to assign a single icon to a panel. Use the <img> HTML tag in the panels' Text to display multiple icons or custom size pictures. The HTMLPicture property adds or replaces a picture in HTML captions.

The following **VB** sample shows "How can I insert an icon to a panel":

```
With StatusBar1
   .BeginUpdate
   .Images
"gBJJgBAIDAAGAAEAAQhYAf8Pf4hh0QihCJo2AEZjQAjEZFEaIEaEEaAIAkcbk0olUrlktl0vmExn
 & _
"/oFBoVDolFo1HpFJpVLplNp1PqFRqVTqlVq1XrFZrVbrldr1fsFhsVjslls1ntFptVrtltt1vuFxuVzul
 & _
"/wGBwWDwmFw2HxGJxWLxmNx0xiFdyOTh8Tf9ZymXx+QytcyNgz8r0OblWjyWds+m0ka1
 & _
"x3GO4NV3WeyvD2XJ5XL5nN51aiw+lfSj0gkUkAEllHanHI5j/cHg8EZf7w8vl8j4f/qfEZeB09/v
 & _
"NAOAEAwCjMBwFAEDwJBMDwLBYAP2/8Hv8/gAGAD8LQs9w/nhDY/oygIA="
   .VisualAppearance.Add 4,"c:\exontrol\images\border.ebn"
   .VisualAppearance.Add 5,"CP:4 1 1 -1 -1"
   .BackColorPanels = 83886080
   .Format = "1,2,3,4,(5/6/7/8)"
   .Panel(1).Image = 1
   .EndUpdate
End With
```

The following **VB.NET** sample shows "How can I insert an icon to a panel":

```
With AxStatusBar1
   .BeginUpdate
   .Images
```

```
"gBJJgBAIDAAGAAEAAQhYAf8Pf4hh0QihCJo2AEZjQAjEZFEaIEaEEaAIAkcbk0oIUrlktl0vmExn
 & _
"/oFBoVDoIFo1HpFJpVLpINp1PqFRqVTqlVq1XrFZrVbrldr1fsFhsVjslls1ntFptVrtltt1vuFxuVzul
 & _
"/wGBwWDwmFw2HxGJxWLxmNx0xiFdyOTh8Tf9ZymXx+QytcyNgz8r0OblWjyWds+m0ka1
 & _
"x3GO4NV3WeyvD2XJ5XL5nN51aiw+IfSj0gkUkAEllHanHI5j/cHg8EZf7w8vl8j4f/qfEZeB09/v
 & _
"NAOAEAwCjMBwFAEDwJBMDwLBYAP2/8Hv8/gAGAD8LQs9w/nhDY/oygIA="
   .VisualAppearance.Add 4,"c:\exontrol\images\border.ebn"
   .VisualAppearance.Add 5,"CP:4 1 1 -1 -1"
   .GetOcx().BackColorPanels = &H5000000
   .Format = "1,2,3,4,(5/6/7/8)"
   .get_Panel(1).Image = 1
   .EndUpdate
End With
```

The following **C++** sample shows "How can I insert an icon to a panel":

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

   #import "D:\\Exontrol\\ExStatusBar\\project\\Debug\\ExStatusBar.dll"
   using namespace EXSTATUSBARLib;
*/
EXSTATUSBARLib::IStatusBarPtr spStatusBar1 = GetDlgItem(IDC_STATUSBAR1)-
>GetControlUnknown();
spStatusBar1->BeginUpdate();
spStatusBar1-
>Images(_bstr_t("gBJJgBAIDAAGAAEAAQhYAf8Pf4hh0QihCJo2AEZjQAjEZFEaIEaEEaAIAkcbk
 +
"/oFBoVDoIFo1HpFJpVLpINp1PqFRqVTqlVq1XrFZrVbrldr1fsFhsVjslls1ntFptVrtltt1vuFxuVzul
 +
"/wGBwWDwmFw2HxGJxWLxmNx0xiFdyOTh8Tf9ZymXx+QytcyNgz8r0OblWjyWds+m0ka1
 +
```

```
"x3GO4NV3WeyvD2XJ5XL5nN51aiw+lfSj0gkUkAEllHanHI5j/cHg8EZf7w8vl8j4f/qfEZeB09/v
 +
"NAOAEAwCjMBwFAEDwJBMDwLBYAP2/8Hv8/gAGAD8LQs9w/nhDY/oygIA=");
spStatusBar1->GetVisualAppearance()->Add(4,"c:\\exontrol\\images\\border.ebn");
spStatusBar1->GetVisualAppearance()->Add(5,"CP:4 1 1 -1 -1");
spStatusBar1->PutBackColorPanels(83886080);
spStatusBar1->PutFormat(L"1,2,3,4,(5/6/7/8)");
spStatusBar1->GetPanel(long(1))->PutImage(1);
spStatusBar1->EndUpdate();
```

The following **C#** sample shows "How can I insert an icon to a panel":

```
axStatusBar1.BeginUpdate();
axStatusBar1.Images("gBJJgBAIDAAGAAEAAQhYAf8Pf4hh0QihCJo2AEZjQAjEZFEaIEaEEaAIA
 +
"/oFBoVDoIFo1HpFJpVLpINp1PqFRqVTqIVq1XrFZrVbrIdr1fsFhsVjslls1ntFptVrtltt1vuFxuVzul
 +
"/wGBwWDwmFw2HxGJxWLxmNx0xiFdyOTh8Tf9ZymXx+QytcyNgz8r0OblWjyWds+m0ka1
 +
"x3GO4NV3WeyvD2XJ5XL5nN51aiw+lfSj0gkUkAEllHanHI5j/cHg8EZf7w8vl8j4f/qfEZeB09/v
 +
"NAOAEAwCjMBwFAEDwJBMDwLBYAP2/8Hv8/gAGAD8LQs9w/nhDY/oygIA=");
axStatusBar1.VisualAppearance.Add(4,"c:\\exontrol\\images\\border.ebn");
axStatusBar1.VisualAppearance.Add(5,"CP:4 1 1 -1 -1");
(axStatusBar1.GetOcx() as EXSTATUSBARLib.StatusBar).BackColorPanels = 0x5000000;
axStatusBar1.Format = "1,2,3,4,(5/6/7/8)";
axStatusBar1.get_Panel(1).Image = 1;
axStatusBar1.EndUpdate();
```

The following **VFP** sample shows "How can I insert an icon to a panel":

```
with thisform.StatusBar1
   .BeginUpdate
   var_s =
"gBJJgBAIDAAGAAEAAQhYAf8Pf4hh0QihCJo2AEZjQAjEZFEaIEaEEaAIAkcbk0oIUrlktl0vmExn

   var_s = var_s +
"oFBoVDoIFo1HpFJpVLpINp1PqFRqVTqIVq1XrFZrVbrIdr1fsFhsVjslls1ntFptVrtltt1vuFxuVzul1
```

```
   var_s = var_s +
"wGBwWDwmFw2HxGJxWLxmNx0xiFdyOTh8Tf9ZymXx+QytcyNgz8r0OblWjyWds+m0ka1V

   var_s = var_s +
"3GO4NV3WeyvD2XJ5XL5nN51aiw+lfSj0gkUkAEllHanHI5j/cHg8EZf7w8vl8j4f/qfEZeB09/vj

   var_s = var_s +
"AOAEAwCjMBwFAEDwJBMDwLBYAP2/8Hv8/gAGAD8LQs9w/nhDY/oygIA="
   .Images(var_s)
   .VisualAppearance.Add(4,"c:\exontrol\images\border.ebn")
   .VisualAppearance.Add(5,"CP:4 1 1 -1 -1")
   .BackColorPanels = 83886080
   .Format = "1,2,3,4,(5/6/7/8)"
   .Panel(1).Image = 1
   .EndUpdate
endwith
```

property StatusBar.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 StatusBar.Panel (Index as Variant) as Panel

Retrieves the panel in the control giving its identifier.

Type	Description
Index as Variant	A Long expression that specifies the index of the panel being accessed.
Panel	A Panel object being accessed

Use the Panel property to access a panel in your status bar control. Use the Format property to add and arrange the panels in the status bar. Use the Text property to specify the caption in the panel. Use the Debug property to display the identifiers of the panels in the status bar. *Use the BackColorPanels property to specify the visual appearance of the borders around the panels.* Use the PanelFromPoint property to get the panel from the cursor.

The following **VB** sample shows how can I add two panels to the status bar control:

```
With StatusBar1
   .BeginUpdate
   .VisualAppearance.Add 4,"c:\exontrol\images\border.ebn"
   .VisualAppearance.Add 5,"CP:4 1 1 -1 -1"
   .BackColorPanels = 83886080
   .Format = "1,2"
   .Panel(1).Text = "Panel 1"
   .Panel(2).Text = "Panel 2"
   .EndUpdate
End With
```

The following **VB** sample shows how can I add three panels aligned from top to bottom:

```
With StatusBar1
   .BeginUpdate
   .BackColor = -2147483633
   .Images
"gBJJgBAIDAAGAAEAAQhYAf8Pf4hh0QihCJo2AEZjQAjEZFEaIEaEEaAIAkcbk0oIUrlktl0vmExn
 & _
"/oFBoVDoIFo1HpFJpVLpINp1PqFRqVTqIVq1XrFZrVbrIdr1fsFhsVjslls1ntFptVrtltt1vuFxuVzul
 & _
"/wGBwWDwmFw2HxGJxWLxmNx0xiFdyOTh8Tf9ZymXx+QytcyNgz8r0OblWjyWds+m0ka1
```

```
 & _
"x3GO4NV3WeyvD2XJ5XL5nN51aiw+lfSj0gkUkAEllHanHI5j/cHg8EZf7w8vl8j4f/qfEZeB09/v
 & _
"NAOAEAwCjMBwFAEDwJBMDwLBYAP2/8Hv8/gAGAD8LQs9w/nhDY/oygIA="
   .VisualAppearance.Add 4,"c:\exontrol\images\border.ebn"
   .VisualAppearance.Add 5,"CP:4 1 1 -1 -1"
   .BackColorPanels = 83886080
   .Format = "1/2/3"
   .Panel(1).Text = "Panel 1"
   .Panel(2).Text = "Panel 2"
   .Panel(3).Text = "Panel 3"
   .EndUpdate
End With
```

The following **VB.NET** sample shows how can I add two panels to the status bar control:

```
With AxStatusBar1
   .BeginUpdate
   .VisualAppearance.Add 4,"c:\exontrol\images\border.ebn"
   .VisualAppearance.Add 5,"CP:4 1 1 -1 -1"
   .GetOcx().BackColorPanels = &H5000000
   .Format = "1,2"
   .get_Panel(1).Text = "Panel 1"
   .get_Panel(2).Text = "Panel 2"
   .EndUpdate
End With
```

The following **VB.NET** sample shows how can I add three panels aligned from top to bottom:

```
With AxStatusBar1
   .BeginUpdate
   .GetOcx().BackColor = &H8000000f
   .Images
"gBJJgBAIDAAGAAEAAQhYAf8Pf4hh0QihCJo2AEZjQAjEZFEaIEaEEaAIAkcbk0oIUrlktl0vmExn
 & _
"/oFBoVDoIFo1HpFJpVLplNp1PqFRqVTqlVq1XrFZrVbrldr1fsFhsVjslls1ntFptVrtltt1vuFxuVzul
 & _
```

```
"/wGBwWDwmFw2HxGJxWLxmNx0xiFdyOTh8Tf9ZymXx+QytcyNgz8r0OblWjyWds+m0ka1
 & _
"x3GO4NV3WeyvD2XJ5XL5nN51aiw+lfSj0gkUkAEllHanHI5j/cHg8EZf7w8vl8j4f/qfEZeB09/v
 & _
"NAOAEAwCjMBwFAEDwJBMDwLBYAP2/8Hv8/gAGAD8LQs9w/nhDY/oygIA="
   .VisualAppearance.Add 4,"c:\exontrol\images\border.ebn"
   .VisualAppearance.Add 5,"CP:4 1 1 -1 -1"
   .GetOcx().BackColorPanels = &H5000000
   .Format = "1/2/3"
   .get_Panel(1).Text = "Panel 1"
   .get_Panel(2).Text = "Panel 2"
   .get_Panel(3).Text = "Panel 3"
   .EndUpdate
End With
```

The following **C++** sample shows how can I add two panels to the status bar control:

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

   #import "D:\\Exontrol\\ExStatusBar\\project\\Debug\\ExStatusBar.dll"
   using namespace EXSTATUSBARLib;
*/
EXSTATUSBARLib::IStatusBarPtr spStatusBar1 = GetDlgItem(IDC_STATUSBAR1)-
>GetControlUnknown();
spStatusBar1->BeginUpdate();
spStatusBar1->GetVisualAppearance()->Add(4,"c:\\exontrol\\images\\border.ebn");
spStatusBar1->GetVisualAppearance()->Add(5,"CP:4 1 1 -1 -1");
spStatusBar1->PutBackColorPanels(83886080);
spStatusBar1->PutFormat(L"1,2");
spStatusBar1->GetPanel(long(1))->PutText(L"Panel 1");
spStatusBar1->GetPanel(long(2))->PutText(L"Panel 2");
spStatusBar1->EndUpdate();
```

The following **C++** sample shows how can I add three panels aligned from top to bottom:

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

   #import "D:\\Exontrol\\ExStatusBar\\project\\Debug\\ExStatusBar.dll"
   using namespace EXSTATUSBARLib;
*/
EXSTATUSBARLib::IStatusBarPtr spStatusBar1 = GetDlgItem(IDC_STATUSBAR1)-
>GetControlUnknown();
spStatusBar1->BeginUpdate();
spStatusBar1->PutBackColor(-2147483633);
spStatusBar1-
>Images(_bstr_t("gBJJgBAIDAAGAAEAAQhYAf8Pf4hh0QihCJo2AEZjQAjEZFEaIEaEEaAIAkcbk
 +
"/oFBoVDolFo1HpFJpVLplNp1PqFRqVTqlVq1XrFZrVbrldr1fsFhsVjslls1ntFptVrtltt1vuFxuVzul
 +
"/wGBwWDwmFw2HxGJxWLxmNx0xiFdyOTh8Tf9ZymXx+QytcyNgz8r0OblWjyWds+m0ka1
 +
"x3GO4NV3WeyvD2XJ5XL5nN51aiw+lfSj0gkUkAEllHanHI5j/cHg8EZf7w8vl8j4f/qfEZeB09/v
 +
"NAOAEAwCjMBwFAEDwJBMDwLBYAP2/8Hv8/gAGAD8LQs9w/nhDY/oygIA=");
spStatusBar1->GetVisualAppearance()->Add(4,"c:\\exontrol\\images\\border.ebn");
spStatusBar1->GetVisualAppearance()->Add(5,"CP:4 1 1 -1 -1");
spStatusBar1->PutBackColorPanels(83886080);
spStatusBar1->PutFormat(L"1/2/3");
spStatusBar1->GetPanel(long(1))->PutText(L"Panel 1");
spStatusBar1->GetPanel(long(2))->PutText(L"Panel 2");
spStatusBar1->GetPanel(long(3))->PutText(L"Panel 3");
spStatusBar1->EndUpdate();
```

The following **C#** sample shows how can I add two panels to the status bar control:

```
axStatusBar1.BeginUpdate();
axStatusBar1.VisualAppearance.Add(4,"c:\\exontrol\\images\\border.ebn");
axStatusBar1.VisualAppearance.Add(5,"CP:4 1 1 -1 -1");
(axStatusBar1.GetOcx() as EXSTATUSBARLib.StatusBar).BackColorPanels = 0x5000000;
```

```
axStatusBar1.Format = "1,2";
axStatusBar1.get_Panel(1).Text = "Panel 1";
axStatusBar1.get_Panel(2).Text = "Panel 2";
axStatusBar1.EndUpdate();
```

The following **C#** sample shows how can I add three panels aligned from top to bottom:

```
axStatusBar1.BeginUpdate();
(axStatusBar1.GetOcx() as EXSTATUSBARLib.StatusBar).BackColor = 0x8000000f;
axStatusBar1.Images("gBJJgBAIDAAGAAEAAQhYAf8Pf4hh0QihCJo2AEZjQAjEZFEaIEaEEaAIA
 +
"/oFBoVDoIFo1HpFJpVLpINp1PqFRqVTqIVq1XrFZrVbrIdr1fsFhsVjslls1ntFptVrtltt1vuFxuVzul
 +
"/wGBwWDwmFw2HxGJxWLxmNx0xiFdyOTh8Tf9ZymXx+QytcyNgz8r0OblWjyWds+m0ka1
 +
"x3GO4NV3WeyvD2XJ5XL5nN51aiw+lfSj0gkUkAEllHanHI5j/cHg8EZf7w8vl8j4f/qfEZeB09/v
 +
"NAOAEAwCjMBwFAEDwJBMDwLBYAP2/8Hv8/gAGAD8LQs9w/nhDY/oygIA=");
axStatusBar1.VisualAppearance.Add(4,"c:\\exontrol\\images\\border.ebn");
axStatusBar1.VisualAppearance.Add(5,"CP:4 1 1 -1 -1");
(axStatusBar1.GetOcx() as EXSTATUSBARLib.StatusBar).BackColorPanels = 0x5000000;
axStatusBar1.Format = "1/2/3";
axStatusBar1.get_Panel(1).Text = "Panel 1";
axStatusBar1.get_Panel(2).Text = "Panel 2";
axStatusBar1.get_Panel(3).Text = "Panel 3";
axStatusBar1.EndUpdate();
```

The following **VFP** sample shows how can I add two panels to the status bar control:

```
with thisform.StatusBar1
   .BeginUpdate
   .VisualAppearance.Add(4,"c:\exontrol\images\border.ebn")
   .VisualAppearance.Add(5,"CP:4 1 1 -1 -1")
   .BackColorPanels = 83886080
   .Format = "1,2"
   .Panel(1).Text = "Panel 1"
   .Panel(2).Text = "Panel 2"
   .EndUpdate
```

```
endwith
```

The following **VFP** sample shows how can I add three panels aligned from top to bottom:

```
with thisform.StatusBar1
   .BeginUpdate
   .BackColor = -2147483633
   var_s =
"gBJJgBAIDAAGAAEAAQhYAf8Pf4hh0QihCJo2AEZjQAjEZFEaIEaEEaAIAkcbk0oIUrlktl0vmExn

   var_s = var_s +
"oFBoVDoIFo1HpFJpVLplNp1PqFRqVTqlVq1XrFZrVbrldr1fsFhsVjslls1ntFptVrtltt1vuFxuVzul1

   var_s = var_s +
"wGBwWDwmFw2HxGJxWLxmNx0xiFdyOTh8Tf9ZymXx+QytcyNgz8r0OblWjyWds+m0ka1V

   var_s = var_s +
"3GO4NV3WeyvD2XJ5XL5nN51aiw+lfSj0gkUkAEllHanHI5j/cHg8EZf7w8vl8j4f/qfEZeB09/vj

   var_s = var_s +
"AOAEAwCjMBwFAEDwJBMDwLBYAP2/8Hv8/gAGAD8LQs9w/nhDY/oygIA="
   .Images(var_s)
   .VisualAppearance.Add(4,"c:\exontrol\images\border.ebn")
   .VisualAppearance.Add(5,"CP:4 1 1 -1 -1")
   .BackColorPanels = 83886080
   .Format = "1/2/3"
   .Panel(1).Text = "Panel 1"
   .Panel(2).Text = "Panel 2"
   .Panel(3).Text = "Panel 3"
   .EndUpdate
endwith
```

property StatusBar.PanelFromPoint (X as OLE_XPOS_PIXELS, Y as OLE_YPOS_PIXELS) as Panel

Retrieves the panel from the point.

Type	Description
X as OLE_XPOS_PIXELS	A single that specifies the current X location of the mouse pointer. The x values is always expressed in client coordinates.
Y as OLE_YPOS_PIXELS	A single that specifies the current Y location of the mouse pointer. The y values is always expressed in client coordinates.
Panel	A Panel object that specifies the panel from the cursor or nothing if no panel at the cursor.

Use the PanelFromPoint property to get the Panel from the point specified by the {X,Y}. The X and Y coordinates are expressed in client coordinates, so a conversion must be done in case your coordinates are relative to the screen or to other window. **If the X parameter is -1 and Y parameter is -1 the PanelFromPoint property determines the handle of the Panel from the cursor**. Use the Text property to access the text of the panel. Use the Index property to identify a panel in the status bar.

The following VB sample displays the caption from the cursor:

```
Private Sub StatusBar1_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
   Dim p As EXSTATUSBARLibCtl.Panel
   With StatusBar1
      Set p = .PanelFromPoint(-1, -1)
      If (Not p Is Nothing) Then
         Debug.Print p.Text
      End If
   End With
End Sub
```

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

```
Private Sub AxStatusBar1_MouseMoveEvent(ByVal sender As System.Object, ByVal e As AxEXSTATUSBARLib._IStatusBarEvents_MouseMoveEvent) Handles AxStatusBar1.MouseMoveEvent
```

```
    Dim p As EXSTATUSBARLib.Panel
    With AxStatusBar1
        p = .get_PanelFromPoint(-1, -1)
        If (Not p Is Nothing) Then
            Debug.Print(p.Text)
        End If
    End With
End Sub
```

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

```
private void axStatusBar1_MouseMoveEvent(object sender,
AxEXSTATUSBARLib._IStatusBarEvents_MouseMoveEvent e)
{
    EXSTATUSBARLib.Panel p = axStatusBar1.get_PanelFromPoint(-1, -1);
    if (p != null)
        System.Diagnostics.Debug.WriteLine(p.Text);
}
```

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

```
void OnMouseMoveStatusbar1(short Button, short Shift, long X, long Y)
{
    CPanel panel = m_statusBar.GetPanelFromPoint( -1, -1 );
    if ( panel.m_lpDispatch != NULL )
        OutputDebugString( panel.GetText() );
}
```

The following VFP sample displays the caption from the cursor:

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

with thisform.StatusBar1
    local p
    p = .PanelFromPoint(-1,-1)
    if ( !isnull(p) )
        wait window nowait p.Text
    endif
```

```
endwith
```

property StatusBar.Picture as IPictureDisp

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

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

By default, the control has no picture associated. The control uses the PictureDisplay property to determine how the picture is displayed on the control's background. Use the BackColor property to specify the control's background color. Use the Appearance property to change the visual appearance of the control's border. Use the BackColorPanels property to assign the same visual aspect for all panels in the status bar control.

The following **VB** sample shows "How do I put a picture on the control's left top corner":

```
With StatusBar1
   .BeginUpdate
   .Picture = StatusBar1.ExecuteTemplate("loadpicture(`c:\exontrol\images\zipdisk.gif`)")
   .PictureDisplay = UpperLeft
   .VisualAppearance.Add 4,"c:\exontrol\images\border.ebn"
   .VisualAppearance.Add 5,"CP:4 1 1 -1 -1"
   .BackColorPanels = 83886080
   .Format = "1,2,3,4,(5/6/7/8)"
   .Debug = True
   .EndUpdate
End With
```

The following **VB.NET** sample shows "How do I put a picture on the control's left top corner":

```
With AxStatusBar1
   .BeginUpdate
   .Picture = AxStatusBar1.ExecuteTemplate("loadpicture(`c:\exontrol\images\zipdisk.gif`)")
   .PictureDisplay = EXSTATUSBARLib.PictureDisplayEnum.UpperLeft
   .VisualAppearance.Add 4,"c:\exontrol\images\border.ebn"
   .VisualAppearance.Add 5,"CP:4 1 1 -1 -1"
```

```
    .GetOcx().BackColorPanels = &H5000000
    .Format = "1,2,3,4,(5/6/7/8)"
    .Debug = True
    .EndUpdate
End With
```

The following **C++** sample shows "How do I put a picture on the control's left top corner":

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

    #import "D:\\Exontrol\\ExStatusBar\\project\\Debug\\ExStatusBar.dll"
    using namespace EXSTATUSBARLib;
*/
EXSTATUSBARLib::IStatusBarPtr spStatusBar1 = GetDlgItem(IDC_STATUSBAR1)-
>GetControlUnknown();
spStatusBar1->BeginUpdate();
spStatusBar1->PutPicture(IPictureDispPtr(((IDispatch*)(spStatusBar1-
>ExecuteTemplate("loadpicture(`c:\\exontrol\\images\\zipdisk.gif`)")))));
spStatusBar1->PutPictureDisplay(EXSTATUSBARLib::UpperLeft);
spStatusBar1->GetVisualAppearance()->Add(4,"c:\\exontrol\\images\\border.ebn");
spStatusBar1->GetVisualAppearance()->Add(5,"CP:4 1 1 -1 -1");
spStatusBar1->PutBackColorPanels(83886080);
spStatusBar1->PutFormat(L"1,2,3,4,(5/6/7/8)");
spStatusBar1->PutDebug(VARIANT_TRUE);
spStatusBar1->EndUpdate();
```

The following **C#** sample shows "How do I put a picture on the control's left top corner":

```
axStatusBar1.BeginUpdate();
(axStatusBar1.GetOcx() as EXSTATUSBARLib.StatusBar).Picture =
(axStatusBar1.ExecuteTemplate("loadpicture(`c:\\exontrol\\images\\zipdisk.gif`)") as
stdole.IPictureDisp);
axStatusBar1.PictureDisplay = EXSTATUSBARLib.PictureDisplayEnum.UpperLeft;
axStatusBar1.VisualAppearance.Add(4,"c:\\exontrol\\images\\border.ebn");
axStatusBar1.VisualAppearance.Add(5,"CP:4 1 1 -1 -1");
```

```
(axStatusBar1.GetOcx() as EXSTATUSBARLib.StatusBar).BackColorPanels = 0x5000000;
axStatusBar1.Format = "1,2,3,4,(5/6/7/8)";
axStatusBar1.Debug = true;
axStatusBar1.EndUpdate();
```

The following **VFP** sample shows "How do I put a picture on the control's left top corner":

```
with thisform.StatusBar1
   .BeginUpdate
   .Picture =
thisform.StatusBar1.ExecuteTemplate("loadpicture(`c:\exontrol\images\zipdisk.gif`)")
   .PictureDisplay = 0
   .VisualAppearance.Add(4,"c:\exontrol\images\border.ebn")
   .VisualAppearance.Add(5,"CP:4 1 1 -1 -1")
   .BackColorPanels = 83886080
   .Format = "1,2,3,4,(5/6/7/8)"
   .Debug = .T.
   .EndUpdate
endwith
```

property StatusBar.PictureDisplay as PictureDisplayEnum

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

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

By default, the control has no picture associated. The control uses the PictureDisplay property to determine how the picture is displayed on the control's background. Use the Picture property to layout a picture on the control's background. Use the BackColor property to specify the control's background color. Use the Appearance property to change the visual appearance of the control's border. Use the BackColorPanels property to assign the same visual aspect for all panels in the status bar control.

The following **VB** sample shows "How do I put a picture on the control's left top corner":

```
With StatusBar1
   .BeginUpdate
   .Picture = StatusBar1.ExecuteTemplate("loadpicture(`c:\exontrol\images\zipdisk.gif`)")
   .PictureDisplay = UpperLeft
   .VisualAppearance.Add 4,"c:\exontrol\images\border.ebn"
   .VisualAppearance.Add 5,"CP:4 1 1 -1 -1"
   .BackColorPanels = 83886080
   .Format = "1,2,3,4,(5/6/7/8)"
   .Debug = True
   .EndUpdate
End With
```

The following **VB.NET** sample shows "How do I put a picture on the control's left top corner":

```
With AxStatusBar1
   .BeginUpdate
   .Picture = AxStatusBar1.ExecuteTemplate("loadpicture(`c:\exontrol\images\zipdisk.gif`)")
   .PictureDisplay = EXSTATUSBARLib.PictureDisplayEnum.UpperLeft
```

```
    .VisualAppearance.Add 4,"c:\exontrol\images\border.ebn"
    .VisualAppearance.Add 5,"CP:4 1 1 -1 -1"
    .GetOcx().BackColorPanels = &H5000000
    .Format = "1,2,3,4,(5/6/7/8)"
    .Debug = True
    .EndUpdate
End With
```

The following **C++** sample shows "How do I put a picture on the control's left top corner":

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

    #import "D:\\Exontrol\\ExStatusBar\\project\\Debug\\ExStatusBar.dll"
    using namespace EXSTATUSBARLib;
*/
EXSTATUSBARLib::IStatusBarPtr spStatusBar1 = GetDlgItem(IDC_STATUSBAR1)-
>GetControlUnknown();
spStatusBar1->BeginUpdate();
spStatusBar1->PutPicture(IPictureDispPtr(((IDispatch*)(spStatusBar1-
>ExecuteTemplate("loadpicture(`c:\\exontrol\\images\\zipdisk.gif`)")))));
spStatusBar1->PutPictureDisplay(EXSTATUSBARLib::UpperLeft);
spStatusBar1->GetVisualAppearance()->Add(4,"c:\\exontrol\\images\\border.ebn");
spStatusBar1->GetVisualAppearance()->Add(5,"CP:4 1 1 -1 -1");
spStatusBar1->PutBackColorPanels(83886080);
spStatusBar1->PutFormat(L"1,2,3,4,(5/6/7/8)");
spStatusBar1->PutDebug(VARIANT_TRUE);
spStatusBar1->EndUpdate();
```

The following **C#** sample shows "How do I put a picture on the control's left top corner":

```
axStatusBar1.BeginUpdate();
(axStatusBar1.GetOcx() as EXSTATUSBARLib.StatusBar).Picture =
(axStatusBar1.ExecuteTemplate("loadpicture(`c:\\exontrol\\images\\zipdisk.gif`)") as
stdole.IPictureDisp);
axStatusBar1.PictureDisplay = EXSTATUSBARLib.PictureDisplayEnum.UpperLeft;
```

```
axStatusBar1.VisualAppearance.Add(4,"c:\\exontrol\\images\\border.ebn");
axStatusBar1.VisualAppearance.Add(5,"CP:4 1 1 -1 -1");
(axStatusBar1.GetOcx() as EXSTATUSBARLib.StatusBar).BackColorPanels = 0x5000000;
axStatusBar1.Format = "1,2,3,4,(5/6/7/8)";
axStatusBar1.Debug = true;
axStatusBar1.EndUpdate();
```

The following **VFP** sample shows "How do I put a picture on the control's left top corner":

```
with thisform.StatusBar1
   .BeginUpdate
   .Picture =
thisform.StatusBar1.ExecuteTemplate("loadpicture(`c:\exontrol\images\zipdisk.gif`)")
   .PictureDisplay = 0
   .VisualAppearance.Add(4,"c:\exontrol\images\border.ebn")
   .VisualAppearance.Add(5,"CP:4 1 1 -1 -1")
   .BackColorPanels = 83886080
   .Format = "1,2,3,4,(5/6/7/8)"
   .Debug = .T.
   .EndUpdate
endwith
```

method StatusBar.Refresh ()

Refreshes the control.

Type	Description

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

The following VB sample calls the Refresh method:

```
StatusBar1.Refresh
```

The following C++ sample calls the Refresh method:

```
m_statusBar.Refresh();
```

The following VB.NET sample calls the Refresh method:

```
AxStatusBar1.CtlRefresh()
```

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

The following C# sample calls the Refresh method:

```
axStatusBar1.CtlRefresh();
```

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

The following VFP sample calls the Refresh method:

```
thisform.StatusBar1.Object.Refresh()
```

method StatusBar.ReplaceIcon ([Icon as Variant], [Index as Variant])

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

Type	Description
Icon as Variant	A long expression that indicates the icon's handle.
Index as Variant	A long expression that indicates the index where icon is inserted.

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

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

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

i = ExStatusBar1.ReplaceIcon(LoadPicture("d:\icons\help.ico").Handle), i specifies the index where the icon is added

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

i = ExStatusBar1.ReplaceIcon(LoadPicture("d:\icons\help.ico").Handle, 0), i is zero, so the first icon is replaced.

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

ExStatusBar1.ReplaceIcon 0, i, i specifies the index of icon removed.

The following VB clears the control's icons collection:

ExStatusBar1.ReplaceIcon 0, -1

property StatusBar.ShowImageList as Boolean

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

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

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

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

Shows the specified tooltip at given position.

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

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

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

For instance:

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

to its default position

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

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

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

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

gradient-center

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

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

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

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

property StatusBar.Template as String

Specifies the control's template.

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

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

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

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

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

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

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

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

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

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

The x-script may uses constant expressions as follow:

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

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

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

property StatusBar.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 StatusBar.TemplatePut (NewVal as Variant)

Defines inside variables for the next Template/ExecuteTemplate call.

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

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

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

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

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

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

Columns.Add("Column1").HeaderBackColor = RGB(255,0,0), adds a new column and changes the column's header back color.

The x-script may uses constant expressions as follow:

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

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

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

property StatusBar.TemplateResult as Variant

Gets the result of the last Template call.

Type	Description
Variant	A VARIANT expression that indicates the result of the last Template call. The TemplateResultN property gets the result as number (double expression). The TemplateResultS property gets the result as string.

The TemplateResult, TemplateResultN, TemplateResultS property returns the result of the last Template call, as variant, numeric (double) or as string. The Template property takes a string called x-script, and executes it. For instance, you can use the TemplateDef, Template, TemplateResult or ExecuteTemplate to work with x-script. It is known that programming languages such as **dBASE Plus, XBasic from AlphaFive, Wonderware**, 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.

For instance, the Wonderware does not support parameters for events, or parameters of any event are not defined during the event, so in this case, you require an alternative in order to get the value for these parameters. Let's say the Select event, which has one parameter ID of long type, which indicates the identifier of the item being selected. The EventParam property gets the value for any parameter of a specified event. The same, the EventParam requires parameters so Wonderware won't support it, in this case, the Template and TemplateResult can be used to get the ID parameter of the Select event as follows:

```
DIM id As Message
#exMenu1.Template = "EventParam(0)";
id = #exMenu1.TemplateResultS;
MessageBox(id, "Identifier", 0);
```

This code must be called during the Select event, else the EventParam has no effect.

The Template script (x-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 StatusBar.TemplateResultN as Double

Gets the result of the last Template call, as double.

Type	Description
Double	A Double expression that indicates the result of the last Template call. The TemplateResult property gets the result as variant. The TemplateResultS property gets the result as string.

The TemplateResult, TemplateResultN, TemplateResultS property returns the result of the last Template call, as variant, numeric (double) or as string. The Template property takes a string called x-script, and executes it. For instance, you can use the TemplateDef, Template, TemplateResult or ExecuteTemplate to work with x-script. It is known that programming languages such as **dBASE Plus, XBasic from AlphaFive, Wonderware**, 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.

For instance, the Wonderware does not support parameters for events, or parameters of any event are not defined during the event, so in this case, you require an alternative in order to get the value for these parameters. Let's say the Select event, which has one parameter ID of long type, which indicates the identifier of the item being selected. The EventParam property gets the value for any parameter of a specified event. The same, the EventParam requires parameters so Wonderware won't support it, in this case, the Template and TemplateResult can be used to get the ID parameter of the Select event as follows:

```
DIM id As Message
#exMenu1.Template = "EventParam(0)";
id = #exMenu1.TemplateResultS;
MessageBox(id, "Identifier", 0);
```

This code must be called during the Select event, else the EventParam has no effect.

The Template script (x-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 StatusBar.TemplateResultS as String

Gets the result of the last Template call, as string.

Type	Description
String	A String expression that indicates the result of the last Template call. The TemplateResultN property gets the result as number (double expression). The TemplateResult property gets the result as variant.

The TemplateResult, TemplateResultN, TemplateResultS property returns the result of the last Template call, as variant, numeric (double) or as string. The Template property takes a string called x-script, and executes it. For instance, you can use the TemplateDef, Template, TemplateResult or ExecuteTemplate to work with x-script. It is known that programming languages such as **dBASE Plus, XBasic from AlphaFive, Wonderware**, 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.

For instance, the Wonderware does not support parameters for events, or parameters of any event are not defined during the event, so in this case, you require an alternative in order to get the value for these parameters. Let's say the Select event, which has one parameter ID of long type, which indicates the identifier of the item being selected. The EventParam property gets the value for any parameter of a specified event. The same, the EventParam requires parameters so Wonderware won't support it, in this case, the Template and TemplateResult can be used to get the ID parameter of the Select event as follows:

```
DIM id As Message
#exMenu1.Template = "EventParam(0)";
id = #exMenu1.TemplateResultS;
MessageBox(id, "Identifier", 0);
```

This code must be called during the Select event, else the EventParam has no effect.

The Template script (x-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 StatusBar.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 ToolTipText property to specify the tooltip being shown when the cursor hovers the panel. Use the ShowToolTip method to display a custom tooltip.

property StatusBar.ToolTipFont as IFontDisp

Retrieves or sets the tooltip's font.

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

Use the ToolTipFont property to assign a font for the control's tooltip. The ToolTipPopDelay property specifies the period in ms of time the ToolTip remains visible if the mouse pointer is stationary within a control. Use the ToolTipWidth property to specify the width of the tooltip window. Use the ToolTipText property to specify the tooltip being shown when the cursor hovers the panel.

property StatusBar.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 ToolTipText property to specify the tooltip being shown when the cursor hovers the panel. Use the ShowToolTip method to display a custom tooltip.

property StatusBar.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 ToolTipText property to specify the tooltip being shown when the cursor hovers the panel. Use the ShowToolTip method to display a custom tooltip.

property StatusBar.Version as String

Retrieves the control's version.

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

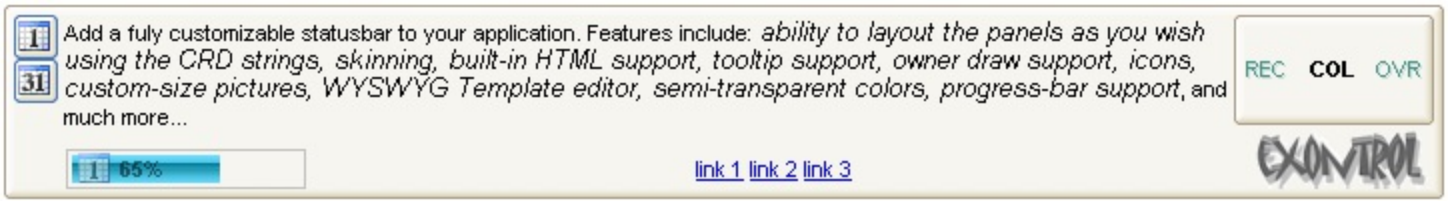
The Version property specifies the control's version.

property StatusBar.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:

- control's **borders** using the Appearance property
- **tooltip** appearance using the Background property
- panel's background using the BackColor property
- background of the panel's percent using the BackColorPercent property
- Any HTML caption that includes an <img> tag.

ExStatusBar 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: {0885027A-DF96-481F-928C-E3E3788889BA}. The object's program identifier is: "Exontrol.StatusBar". The /COM object module is: "ExStatusBar.dll"

The StatusBar component supports the following events:

Name	Description
AnchorClick	Occurs when the anchor element is clicked.
Click	Occurs when the user presses and then releases the left mouse button over the control.
ClickPanel	Occurs when the user clicks a panel.
DblClick	Occurs when the user dblclk the left mouse button over an object.
DblClickPanel	Occurs when the user double clicks a panel.
Event	Notifies the application once the control fires an event.
KeyDown	Occurs when the user presses a key while an object has the focus.
KeyPress	Occurs when the user presses and releases an ANSI key.
KeyUp	Occurs when the user releases a key while an object has the focus.
MouseDown	Occurs when the user presses a mouse button.
MouseMove	Occurs when the user moves the mouse.
MouseUp	Occurs when the user releases a mouse button.
OleEvent	Occurs when an inside ActiveX control fires an event.
OwnerDrawEnd	Ends painting the owner draw panel.
OwnerDrawStart	Starts painting the owner draw panel.
PercentChange	Occurs when the Percent property is changed.

event AnchorClick (AnchorID as String, Options as String)

Occurs when the anchor element is clicked.

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

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

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

C#

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

VB

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

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

C#

```
private void AnchorClick(object sender,
AxEXSTATUSBARLib._IStatusBarEvents_AnchorClickEvent e)
{
}
```

C++

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

C++ Builder

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

Delphi

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

Delphi 8 (.NET only)

```
procedure AnchorClick(sender: System.Object; e:
AxEXSTATUSBARLib._IStatusBarEvents_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
AxEXSTATUSBARLib._IStatusBarEvents_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(oStatusBar,AnchorID,Options)
RETURN
```

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

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

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

Visual Data...
```
Procedure OnComAnchorClick String llAnchorID String llOptions
   Forward Send OnComAnchorClick llAnchorID llOptions
End_Procedure
```

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

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

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

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

The following VB sample displays the identifier of the hyperlink being clicked:

```
Private Sub StatusBar1_AnchorClick(ByVal AnchorID As String, ByVal Options As String)
   Debug.Print AnchorID
End Sub
```

The following VB.NET sample displays the identifier of the hyperlink being clicked:

```
Private Sub AxStatusBar1_AnchorClick(ByVal sender As Object, ByVal e As
AxEXSTATUSBARLib._IStatusBarEvents_AnchorClickEvent) Handles
AxStatusBar1.AnchorClick
```

```
    Debug.Print(e.anchorID)
End Sub
```

The following C# sample displays the identifier of the hyperlink being clicked:

```
private void axStatusBar1_AnchorClick(object sender,
AxEXSTATUSBARLib._IStatusBarEvents_AnchorClickEvent e)
{
    System.Diagnostics.Debug.WriteLine(e.anchorID);
}
```

The following C++ sample displays the identifier of the hyperlink being clicked:

```
void OnAnchorClickStatusbar1(LPCTSTR AnchorID, LPCTSTR Options)
{
    MessageBox( AnchorID );
}
```

The following VFP sample displays the identifier of the hyperlink being clicked:

```
*** ActiveX Control Event ***
LPARAMETERS panel

with panel
    .Text = LTrim(RTrim(.Percent)) + "%"
endwith
```

event Click ()

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

Type	Description

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

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

C#

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

VB

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

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

C#

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

C++

```
void OnClick()
{
}
```

C++ Builder

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

Delphi

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

Delphi 8 (.NET only)

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

Powe…

```
begin event Click()
end event Click
```

VB.NET

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

VB6

```
Private Sub Click()
End Sub
```

VBA

```
Private Sub Click()
End Sub
```

VFP

```
LPARAMETERS nop
```

Xbas…

```
PROCEDURE OnClick(oStatusBar)
RETURN
```

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

Java…

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

VBSc…

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

Visual Data…

```
Procedure OnComClick
   Forward Send OnComClick
```

```
End_Procedure
```

Visual Objects

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

X++

```
void onEvent_Click()
{
}
```

XBasic

```
function Click as v ()
end function
```

dBASE

```
function nativeObject_Click()
return
```

event ClickPanel (Panel as Panel)

Occurs when the user clicks a panel.

Type	Description
Panel as Panel	A Panel object being clicked.

The ClickPanel event notifies your application when the user clicks a panel in the status bar control. The ClickPanel event is not fired if a panel is disabled. Use the Enabled property to enable or disable a panel. If the panel is disabled or the user presses the right mouse button, you still can find out the panel from the cursor using the PanelFromPoint property. The ClickPanel event is fired when the user releases the left mouse button over the panel. Use the DblClickPanel event to notify your application whether the user double clicked a panel.

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

C#

```
private void ClickPanel(object sender,exontrol.EXSTATUSBARLib.Panel Panel)
{
}
```

VB

```
Private Sub ClickPanel(ByVal sender As System.Object,ByVal Panel As
exontrol.EXSTATUSBARLib.Panel) Handles ClickPanel
End Sub
```

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

C#

```
private void ClickPanel(object sender,
AxEXSTATUSBARLib._IStatusBarEvents_ClickPanelEvent e)
{
}
```

C++

```
void OnClickPanel(LPDISPATCH Panel)
{
}
```

C++ Builder

```
void __fastcall ClickPanel(TObject *Sender,Exstatusbarlib_tlb::IPanel *Panel)
{
}
```

Delphi

```
procedure ClickPanel(ASender: TObject; Panel : IPanel);
```

```
begin
end;
```

Delphi 8 (.NET only)

```
procedure ClickPanel(sender: System.Object; e:
AxEXSTATUSBARLib._IStatusBarEvents_ClickPanelEvent);
begin
end;
```

Powe…

```
begin event ClickPanel(oleobject Panel)
end event ClickPanel
```

VB.NET

```
Private Sub ClickPanel(ByVal sender As System.Object, ByVal e As
AxEXSTATUSBARLib._IStatusBarEvents_ClickPanelEvent) Handles ClickPanel
End Sub
```

VB6

```
Private Sub ClickPanel(ByVal Panel As EXSTATUSBARLibCtl.IPanel)
End Sub
```

VBA

```
Private Sub ClickPanel(ByVal Panel As Object)
End Sub
```

VFP

```
LPARAMETERS Panel
```

Xbas…

```
PROCEDURE OnClickPanel(oStatusBar,Panel)
RETURN
```

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

Java…

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

VBSc…

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

Visual Data…

```
Procedure OnComClickPanel Variant llPanel
   Forward Send OnComClickPanel llPanel
End_Procedure
```

Visual Objects

```
METHOD OCX_ClickPanel(Panel) CLASS MainDialog
RETURN NIL
```

X++

```
void onEvent_ClickPanel(COM _Panel)
{
}
```

XBasic

```
function ClickPanel as v (Panel as OLE::Exontrol.StatusBar.1::IPanel)
end function
```

dBASE

```
function nativeObject_ClickPanel(Panel)
return
```

event DblClick (Shift as Integer, X as OLE_XPOS_PIXELS, Y as OLE_YPOS_PIXELS)

Occurs when the user dblclk the left mouse button over an object.

Type	Description
Shift as Integer	An integer that corresponds to the state of the SHIFT, CTRL, and ALT keys.
X as OLE_XPOS_PIXELS	A single that specifies the current X location of the mouse pointer. The x values is always expressed in container coordinates.
Y as OLE_YPOS_PIXELS	A single that specifies the current Y location of the mouse pointer. The y values is always expressed in container coordinates

The DblClick event is fired when user double clicks the control. Use the PanelFromPoint method to determine the cell over the cursor. The following samples display the caption of the panel being double clicked. Use the Text property to specify the panel's caption.

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

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

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

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

C#
```
private void DblClick(object sender,
AxEXSTATUSBARLib._IStatusBarEvents_DblClickEvent e)
{
}
```

C++
```
void OnDblClick(short Shift,long X,long Y)
{
}
```

C++ Builder

```
void __fastcall DblClick(TObject *Sender,short Shift,int X,int Y)
{
}
```

Delphi

```
procedure DblClick(ASender: TObject; Shift : Smallint;X : Integer;Y : Integer);
begin
end;
```

Delphi 8 (.NET only)

```
procedure DblClick(sender: System.Object; e:
AxEXSTATUSBARLib._IStatusBarEvents_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
AxEXSTATUSBARLib._IStatusBarEvents_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(oStatusBar,Shift,X,Y)
RETURN
```

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

Java…

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

VBSc…

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

Visual Data…

```
Procedure OnComDblClick Short llShift OLE_XPOS_PIXELS llX OLE_YPOS_PIXELS llY
   Forward Send OnComDblClick llShift llX llY
End_Procedure
```

Visual Objects

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

X++

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

XBasic

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

dBASE

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

The following VB sample displays the caption of the panel being double clicked:

```
Private Sub StatusBar1_DblClick(Shift As Integer, X As Single, Y As Single)
   Dim p As EXSTATUSBARLibCtl.Panel
   With StatusBar1
      Set p = .PanelFromPoint(-1, -1)
      If (Not p Is Nothing) Then
         Debug.Print p.Text
      End If
   End With
End Sub
```

The following VB.NET sample displays the caption of the panel being double clicked:

```
Private Sub AxStatusBar1_DblClick(ByVal sender As System.Object, ByVal e As
AxEXSTATUSBARLib._IStatusBarEvents_DblClickEvent) Handles AxStatusBar1.DblClick
   Dim p As EXSTATUSBARLib.Panel
   With AxStatusBar1
      p = .get_PanelFromPoint(-1, -1)
      If (Not p Is Nothing) Then
         Debug.Print(p.Text)
      End If
   End With
End Sub
```

The following C# sample displays the caption of the panel being double clicked:

```
private void axStatusBar1_DblClick(object sender,
AxEXSTATUSBARLib._IStatusBarEvents_DblClickEvent e)
{
   EXSTATUSBARLib.Panel p = axStatusBar1.get_PanelFromPoint(-1, -1);
   if (p != null)
      System.Diagnostics.Debug.WriteLine(p.Text);
}
```

The following C++ sample displays the caption of the panel being double clicked:

```
void OnDblClickStatusbar1(short Shift, long X, long Y)
{
   CPanel panel = m_statusBar.GetPanelFromPoint( -1, -1 );
   if ( panel.m_lpDispatch != NULL )
      OutputDebugString( panel.GetText() );
}
```

The following VFP sample displays the caption of the panel being double clicked:

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

with thisform.StatusBar1
   local p
   p = .PanelFromPoint(-1,-1)
```

```
   if ( !isnull(p) )
      wait window nowait p.Text
   endif
endwith
```

event DblClickPanel (Panel as Panel)

Occurs when the user double clicks a panel.

Type	Description
Panel as Panel	A Panel object being double clicked.

The DblClickPanel event notifies your application when the user double clicks a panel in the status bar control. The DblClickPanel event is not fired if a panel is disabled. Use the Enabled property to enable or disable a panel. If the panel is disabled or the user presses the right mouse button, you still can find out the panel from the cursor using the PanelFromPoint property. The DblClickPanel event is fired when the user releases the left mouse button over the panel. Use the ClickPanel event to notify your application whether the user clicks a panel.

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

C#

```
private void DblClickPanel(object sender,exontrol.EXSTATUSBARLib.Panel Panel)
{
}
```

VB

```
Private Sub DblClickPanel(ByVal sender As System.Object,ByVal Panel As
exontrol.EXSTATUSBARLib.Panel) Handles DblClickPanel
End Sub
```

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

C#

```
private void DblClickPanel(object sender,
AxEXSTATUSBARLib._IStatusBarEvents_DblClickPanelEvent e)
{
}
```

C++

```
void OnDblClickPanel(LPDISPATCH Panel)
{
}
```

C++ Builder

```
void __fastcall DblClickPanel(TObject *Sender,Exstatusbarlib_tlb::IPanel *Panel)
{
}
```

Delphi

```
procedure DblClickPanel(ASender: TObject; Panel : IPanel);
```

```
begin
end;
```

Delphi 8 (.NET only)

```
procedure DblClickPanel(sender: System.Object; e:
AxEXSTATUSBARLib._IStatusBarEvents_DblClickPanelEvent);
begin
end;
```

Powe…

```
begin event DblClickPanel(oleobject Panel)
end event DblClickPanel
```

VB.NET

```
Private Sub DblClickPanel(ByVal sender As System.Object, ByVal e As
AxEXSTATUSBARLib._IStatusBarEvents_DblClickPanelEvent) Handles DblClickPanel
End Sub
```

VB6

```
Private Sub DblClickPanel(ByVal Panel As EXSTATUSBARLibCtl.IPanel)
End Sub
```

VBA

```
Private Sub DblClickPanel(ByVal Panel As Object)
End Sub
```

VFP

```
LPARAMETERS Panel
```

Xbas…

```
PROCEDURE OnDblClickPanel(oStatusBar,Panel)
RETURN
```

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

Java…

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

VBSc…

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

Visual Data…

```
Procedure OnComDblClickPanel Variant llPanel
   Forward Send OnComDblClickPanel llPanel
End_Procedure
```

Visual Objects

```
METHOD OCX_DblClickPanel(Panel) CLASS MainDialog
RETURN NIL
```

X++

```
void onEvent_DblClickPanel(COM _Panel)
{
}
```

XBasic

```
function DblClickPanel as v (Panel as OLE::Exontrol.StatusBar.1::IPanel)
end function
```

dBASE

```
function nativeObject_DblClickPanel(Panel)
return
```

event Event (EventID as Long)

Notifies the application once the control fires an event.

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

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

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

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

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

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

```
// Notifies the application once the control fires an event.
void onEvent_Event(int _EventID)
{
   print exstatusbar1.EventParam(-2).toString();
}
```

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

```
"MouseMove/-606( 1 , 0 , 145 , 36 )" VT_BSTR
"BarParentChange/125( 192998632 , 'B' , 192999592 , =false )" VT_BSTR
"BeforeDrawPart/54( 2 , -1962866148 , =0 , =0 , =0 , =0 , =false )" VT_BSTR
```

```
"AfterDrawPart/55( 2 , -1962866148 , 0 , 0 , 0 , 0 )" VT_BSTR
"MouseMove/-606( 1 , 0 , 145 , 35 )" VT_BSTR
```

Each line indicates an event, and the following information is provided: the name of the event, its identifier, and the list of parameters being passed to the event. The parameters that starts with = character, indicates a parameter by reference, in other words one that can changed during the event handler.

Now, we can see that the identifier for the BarParentChange event is 125, so we need to handle the Event event as:

```
// Notifies the application once the control fires an event.
void onEvent_Event(int _EventID)
{
   ;
   if ( _EventID == 125 ) /*event BarParentChange (Item as HITEM, Key as Variant, NewItem as HITEM, Cancel as Boolean) */
      exstatusbar1.EventParam( 3 /*Cancel*/, COMVariant::createFromBoolean(true) );
}
```

The code checks if the BarParentChange (_EventID == 125) event is fired, and changes the third parameter of the event to true. The definition for BarParentChange event can be consulted in the control's documentation or in the ActiveX explorer. So, anytime you need to access the original parameters for the event you should use the EventParam method that allows you to get or set a parameter. If the parameter is not passed by reference, you can not change the parameter's value.

Now, let's add some code to see a complex sample, so let's say that we need to prevent moving the bar from an item to any disabled item. So, we need to specify the Cancel parameter as not Items.EnableItem(NewItem), in other words cancels if the new parent is disabled. Shortly the code will be:

```
// Notifies the application once the control fires an event.
void onEvent_Event(int _EventID)
{
   ;
   if ( _EventID == 125 ) /*event BarParentChange (Item as HITEM, Key as Variant, NewItem as HITEM, Cancel as Boolean) */
      if ( !exstatusbar1.Items().EnableItem( exstatusbar1.EventParam( 2 /*NewItem*/ ) ) )
         exstatusbar1.EventParam( 3 /*Cancel*/, COMVariant::createFromBoolean(true) );
}
```

In conclusion, anytime the X++ fires the "invalid parameters." while handling an event, you can use and handle the Event notification and EventParam methods of the control

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

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

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

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

C#
```
private void Event(object sender,
AxEXSTATUSBARLib._IStatusBarEvents_EventEvent e)
{
}
```

C++
```
void OnEvent(long EventID)
{
}
```

C++ Builder
```
void __fastcall Event(TObject *Sender,long EventID)
{
}
```

Delphi
```
procedure Event(ASender: TObject; EventID : Integer);
begin
end;
```

Delphi 8 (.NET only)
```
procedure Event(sender: System.Object; e:
AxEXSTATUSBARLib._IStatusBarEvents_EventEvent);
begin
end;
```

Powe…
```
begin event Event(long EventID)
end event Event
```

VB.NET

```
Private Sub Event(ByVal sender As System.Object, ByVal e As
AxEXSTATUSBARLib._IStatusBarEvents_EventEvent) Handles Event
End Sub
```

VB6

```
Private Sub Event(ByVal EventID As Long)
End Sub
```

VBA

```
Private Sub Event(ByVal EventID As Long)
End Sub
```

VFP

```
LPARAMETERS EventID
```

Xbas…

```
PROCEDURE OnEvent(oStatusBar,EventID)
RETURN
```

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

Java…

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

VBSc…

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

Visual Data…

```
Procedure OnComEvent Integer llEventID
   Forward Send OnComEvent llEventID
End_Procedure
```

Visual Objects

```
METHOD OCX_Event(EventID) CLASS MainDialog
RETURN NIL
```

X++

```
void onEvent_Event(int _EventID)
{
}
```

XBasic

```
function Event as v (EventID as N)
end function
```

dBASE

```
function nativeObject_Event(EventID)
return
```

event KeyDown (KeyCode as Integer, Shift as Integer)

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

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

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

ShiftDown = (Shift And 1) > 0
CtrlDown = (Shift And 2) > 0
AltDown = (Shift And 4) > 0
In a procedure, you can test for any combination of conditions, as in this example:
If AltDown And CtrlDown Then

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

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

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

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

C#
```
private void KeyDownEvent(object sender,
AxEXSTATUSBARLib._IStatusBarEvents_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:
AxEXSTATUSBARLib._IStatusBarEvents_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
AxEXSTATUSBARLib._IStatusBarEvents_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(oStatusBar,KeyCode,Shift)
RETURN
```

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

Java…
```
<SCRIPT EVENT="KeyDown(KeyCode,Shift)" LANGUAGE="JScript">
</SCRIPT>
```

VBSc…
```
<SCRIPT LANGUAGE="VBScript">
Function KeyDown(KeyCode,Shift)
End Function
</SCRIPT>
```

Visual Data…
```
Procedure OnComKeyDown Short llKeyCode Short llShift
   Forward Send OnComKeyDown llKeyCode llShift
End_Procedure
```

Visual Objects
```
METHOD OCX_KeyDown(KeyCode,Shift) CLASS MainDialog
RETURN NIL
```

X++
```
void onEvent_KeyDown(COMVariant /*short*/ _KeyCode,int _Shift)
{
}
```

XBasic
```
function KeyDown as v (KeyCode as N,Shift as N)
end function
```

dBASE
```
function nativeObject_KeyDown(KeyCode,Shift)
return
```

event KeyPress (KeyAscii as Integer)

Occurs when the user presses and releases an ANSI key.

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

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

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

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

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

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

C#
```
private void KeyPressEvent(object sender,
AxEXSTATUSBARLib._IStatusBarEvents_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:
AxEXSTATUSBARLib._IStatusBarEvents_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
AxEXSTATUSBARLib._IStatusBarEvents_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(oStatusBar,KeyAscii)
RETURN
```

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

Java…

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

VBSc…

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

Visual Data…

```
Procedure OnComKeyPress Short llKeyAscii
    Forward Send OnComKeyPress llKeyAscii
End_Procedure
```

Visual Objects

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

X++

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

XBasic

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

dBASE

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

event KeyUp (KeyCode as Integer, Shift as Integer)

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

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

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

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

C#

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

VB

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

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

C#

```
private void KeyUpEvent(object sender,
AxEXSTATUSBARLib._IStatusBarEvents_KeyUpEvent e)
{
}
```

C++

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

C++ Builder

```
void __fastcall KeyUp(TObject *Sender,short * KeyCode,short Shift)
{
```

```
}
```

Delphi

```
procedure KeyUp(ASender: TObject; var KeyCode : Smallint;Shift : Smallint);
begin
end;
```

Delphi 8 (.NET only)

```
procedure KeyUpEvent(sender: System.Object; e:
AxEXSTATUSBARLib._IStatusBarEvents_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
AxEXSTATUSBARLib._IStatusBarEvents_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(oStatusBar,KeyCode,Shift)
RETURN
```

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

Java…

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

VBSc…

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

```
</SCRIPT>
```

Visual Data…

```
Procedure OnComKeyUp Short llKeyCode Short llShift
    Forward Send OnComKeyUp llKeyCode llShift
End_Procedure
```

Visual Objects

```
METHOD OCX_KeyUp(KeyCode,Shift) CLASS MainDialog
RETURN NIL
```

X++

```
void onEvent_KeyUp(COMVariant /*short*/ _KeyCode,int _Shift)
{
}
```

XBasic

```
function KeyUp as v (KeyCode as N,Shift as N)
end function
```

dBASE

```
function nativeObject_KeyUp(KeyCode,Shift)
return
```

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

Occurs when the user presses a mouse button.

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

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

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

C#

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

VB

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

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

C#

```
private void MouseDownEvent(object sender,
AxEXSTATUSBARLib._IStatusBarEvents_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:
AxEXSTATUSBARLib._IStatusBarEvents_MouseDownEvent);
begin
end;
```

Powe…

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

VB.NET

```
Private Sub MouseDownEvent(ByVal sender As System.Object, ByVal e As
AxEXSTATUSBARLib._IStatusBarEvents_MouseDownEvent) Handles
MouseDownEvent
End Sub
```

VB6

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

VBA

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

VFP

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

Xbas…

```
PROCEDURE OnMouseDown(oStatusBar,Button,Shift,X,Y)
RETURN
```

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

Java…
```
<SCRIPT EVENT="MouseDown(Button,Shift,X,Y)" LANGUAGE="JScript">
</SCRIPT>
```

VBSc…
```
<SCRIPT LANGUAGE="VBScript">
Function MouseDown(Button,Shift,X,Y)
End Function
</SCRIPT>
```

Visual Data…
```
Procedure OnComMouseDown Short llButton Short llShift OLE_XPOS_PIXELS llX
OLE_YPOS_PIXELS llY
   Forward Send OnComMouseDown llButton llShift llX llY
End_Procedure
```

Visual Objects
```
METHOD OCX_MouseDown(Button,Shift,X,Y) CLASS MainDialog
RETURN NIL
```

X++
```
void onEvent_MouseDown(int _Button,int _Shift,int _X,int _Y)
{
}
```

XBasic
```
function MouseDown as v (Button as N,Shift as N,X as
OLE::Exontrol.StatusBar.1::OLE_XPOS_PIXELS,Y as
OLE::Exontrol.StatusBar.1::OLE_YPOS_PIXELS)
end function
```

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

The following VB sample displays the caption of the panel being clicked:

```
Private Sub StatusBar1_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As
Single)
   Dim p As EXSTATUSBARLibCtl.Panel
   With StatusBar1
      Set p = .PanelFromPoint(-1, -1)
```

```
    If (Not p Is Nothing) Then
        Debug.Print p.Text
    End If
  End With
End Sub
```

The following VB.NET sample displays the caption of the panel being clicked:

```
Private Sub AxStatusBar1_MouseDownEvent(ByVal sender As System.Object, ByVal e As
AxEXSTATUSBARLib._IStatusBarEvents_MouseDownEvent) Handles
AxStatusBar1.MouseDownEvent
  Dim p As EXSTATUSBARLib.Panel
  With AxStatusBar1
    p = .get_PanelFromPoint(-1, -1)
    If (Not p Is Nothing) Then
      Debug.Print(p.Text)
    End If
  End With
End Sub
```

The following C# sample displays the caption of the panel being clicked:

```
private void axStatusBar1_MouseDownEvent(object sender,
AxEXSTATUSBARLib._IStatusBarEvents_MouseDownEvent e)
{
  EXSTATUSBARLib.Panel p = axStatusBar1.get_PanelFromPoint(-1, -1);
  if (p != null)
    System.Diagnostics.Debug.WriteLine(p.Text);
}
```

The following C++ sample displays the caption of the panel being clicked:

```
void OnMouseDownStatusbar1(short Button, short Shift, long X, long Y)
{
  CPanel panel = m_statusBar.GetPanelFromPoint( -1, -1 );
  if ( panel.m_lpDispatch != NULL )
    OutputDebugString( panel.GetText() );
}
```

The following VFP sample displays the caption of the panel being clicked:

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

with thisform.StatusBar1
   local p
   p = .PanelFromPoint(-1,-1)
   if ( !isnull(p) )
      wait window nowait p.Text
   endif
endwith
```

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

Occurs when the user moves the mouse.

Type	Description
Button as Integer	An integer that corresponds to the state of the mouse buttons in which a bit is set if the button is down.
Shift as Integer	An integer that corresponds to the state of the SHIFT, CTRL, and ALT keys.
X as OLE_XPOS_PIXELS	A single that specifies the current X location of the mouse pointer. The x values is always expressed in container coordinates.
Y as OLE_YPOS_PIXELS	A single that specifies the current Y location of the mouse pointer. The y values is always expressed in container coordinates.

The MouseMove event is generated continually as the mouse pointer moves across objects. Unless another object has captured the mouse, an object recognizes a MouseMove event whenever the mouse position is within its borders. Use the PanelFromPoint property to get the panel from the cursor.

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

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

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

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

C#
```
private void MouseMoveEvent(object sender,
AxEXSTATUSBARLib._IStatusBarEvents_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:
AxEXSTATUSBARLib._IStatusBarEvents_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
AxEXSTATUSBARLib._IStatusBarEvents_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(oStatusBar,Button,Shift,X,Y)
```

```
RETURN
```

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

Java…

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

VBSc…

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

Visual Data…

```
Procedure OnComMouseMove Short llButton Short llShift OLE_XPOS_PIXELS llX
OLE_YPOS_PIXELS llY
   Forward Send OnComMouseMove llButton llShift llX llY
End_Procedure
```

Visual Objects

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

X++

```
void onEvent_MouseMove(int _Button,int _Shift,int _X,int _Y)
{
}
```

XBasic

```
function MouseMove as v (Button as N,Shift as N,X as
OLE::Exontrol.StatusBar.1::OLE_XPOS_PIXELS,Y as
OLE::Exontrol.StatusBar.1::OLE_YPOS_PIXELS)
end function
```

dBASE

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

The following VB sample displays the caption from the cursor:

```
Private Sub StatusBar1_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As
Single)
   Dim p As EXSTATUSBARLibCtl.Panel
```

```
    With StatusBar1
        Set p = .PanelFromPoint(-1, -1)
        If (Not p Is Nothing) Then
            Debug.Print p.Text
        End If
    End With
End Sub
```

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

```
Private Sub AxStatusBar1_MouseMoveEvent(ByVal sender As System.Object, ByVal e As
AxEXSTATUSBARLib._IStatusBarEvents_MouseMoveEvent) Handles
AxStatusBar1.MouseMoveEvent
    Dim p As EXSTATUSBARLib.Panel
    With AxStatusBar1
        p = .get_PanelFromPoint(-1, -1)
        If (Not p Is Nothing) Then
            Debug.Print(p.Text)
        End If
    End With
End Sub
```

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

```
private void axStatusBar1_MouseMoveEvent(object sender,
AxEXSTATUSBARLib._IStatusBarEvents_MouseMoveEvent e)
{
    EXSTATUSBARLib.Panel p = axStatusBar1.get_PanelFromPoint(-1, -1);
    if (p != null)
        System.Diagnostics.Debug.WriteLine(p.Text);
}
```

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

```
void OnMouseMoveStatusbar1(short Button, short Shift, long X, long Y)
{
    CPanel panel = m_statusBar.GetPanelFromPoint( -1, -1 );
    if ( panel.m_lpDispatch != NULL )
        OutputDebugString( panel.GetText() );
```

```
}
```

The following VFP sample displays the caption from the cursor:

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

with thisform.StatusBar1
   local p
   p = .PanelFromPoint(-1,-1)
   if ( !isnull(p) )
      wait window nowait p.Text
   endif
endwith
```

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

Occurs when the user releases a mouse button.

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

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

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

C#

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

VB

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

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

C#

```
private void MouseUpEvent(object sender,
AxEXSTATUSBARLib._IStatusBarEvents_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:
AxEXSTATUSBARLib._IStatusBarEvents_MouseUpEvent);
begin
end;
```

Powe…

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

VB.NET

```
Private Sub MouseUpEvent(ByVal sender As System.Object, ByVal e As
AxEXSTATUSBARLib._IStatusBarEvents_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(oStatusBar,Button,Shift,X,Y)
RETURN
```

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

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

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

Visual Data...
```
Procedure OnComMouseUp Short llButton Short llShift OLE_XPOS_PIXELS llX
OLE_YPOS_PIXELS llY
   Forward Send OnComMouseUp llButton llShift llX llY
End_Procedure
```

Visual Objects
```
METHOD OCX_MouseUp(Button,Shift,X,Y) CLASS MainDialog
RETURN NIL
```

X++
```
void onEvent_MouseUp(int _Button,int _Shift,int _X,int _Y)
{
}
```

XBasic
```
function MouseUp as v (Button as N,Shift as N,X as
OLE::Exontrol.StatusBar.1::OLE_XPOS_PIXELS,Y as
OLE::Exontrol.StatusBar.1::OLE_YPOS_PIXELS)
end function
```

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

The following VB sample displays the caption of the panel being clicked:

```
Private Sub StatusBar1_MouseUp(Button As Integer, Shift As Integer, X As Single, Y As
Single)
   Dim p As EXSTATUSBARLibCtl.Panel
   With StatusBar1
      Set p = .PanelFromPoint(-1, -1)
```

```
        If (Not p Is Nothing) Then
            Debug.Print p.Text
        End If
    End With
End Sub
```

The following VB.NET sample displays the caption of the panel being clicked:

```
Private Sub AxStatusBar1_MouseUpEvent(ByVal sender As System.Object, ByVal e As
AxEXSTATUSBARLib._IStatusBarEvents_MouseUpEvent) Handles
AxStatusBar1.MouseUpEvent
    Dim p As EXSTATUSBARLib.Panel
    With AxStatusBar1
        p = .get_PanelFromPoint(-1, -1)
        If (Not p Is Nothing) Then
            Debug.Print(p.Text)
        End If
    End With
End Sub
```

The following C# sample displays the caption of the panel being clicked:

```
private void axStatusBar1_MouseUpEvent(object sender,
AxEXSTATUSBARLib._IStatusBarEvents_MouseUpEvent e)
{
    EXSTATUSBARLib.Panel p = axStatusBar1.get_PanelFromPoint(-1, -1);
    if (p != null)
        System.Diagnostics.Debug.WriteLine(p.Text);
}
```

The following C++ sample displays the caption of the panel being clicked:

```
void OnMouseUpStatusbar1(short Button, short Shift, long X, long Y)
{
    CPanel panel = m_statusBar.GetPanelFromPoint( -1, -1 );
    if ( panel.m_lpDispatch != NULL )
        OutputDebugString( panel.GetText() );
}
```

The following VFP sample displays the caption of the panel being clicked:

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

with thisform.StatusBar1
   local p
   p = .PanelFromPoint(-1,-1)
   if ( !isnull(p) )
      wait window nowait p.Text
   endif
endwith
```

event OleEvent (Panel as Panel, Ev as OleEvent)

Occurs when an inside ActiveX control fires an event.

Type	Description
Panel as Panel	A Panel object that hosts an ActiveX.
Ev as OleEvent	An Ev object that holds information about the fired event.

The OleEvent event notifies your application when an inner ActiveX control fires an event. Use the ToString property to display information about fired event. Use the ControlID property to insert an ActiveX control to a panel.

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

C#

```
private void OleEvent(object sender,exontrol.EXSTATUSBARLib.Panel
Panel,exontrol.EXSTATUSBARLib.OleEvent Ev)
{
}
```

VB

```
Private Sub OleEvent(ByVal sender As System.Object,ByVal Panel As
exontrol.EXSTATUSBARLib.Panel,ByVal Ev As exontrol.EXSTATUSBARLib.OleEvent)
Handles OleEvent
End Sub
```

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

C#

```
private void OleEvent(object sender,
AxEXSTATUSBARLib._IStatusBarEvents_OleEventEvent e)
{
}
```

C++

```
void OnOleEvent(LPDISPATCH Panel,LPDISPATCH Ev)
{
}
```

C++ Builder

```
void __fastcall OleEvent(TObject *Sender,Exstatusbarlib_tlb::IPanel
*Panel,Exstatusbarlib_tlb::IOleEvent *Ev)
{
}
```

Delphi

```
procedure OleEvent(ASender: TObject; Panel : IPanel;Ev : IOleEvent);
begin
end;
```

Delphi 8 (.NET only)

```
procedure OleEvent(sender: System.Object; e:
AxEXSTATUSBARLib._IStatusBarEvents_OleEventEvent);
begin
end;
```

Powe…

```
begin event OleEvent(oleobject Panel,oleobject Ev)
end event OleEvent
```

VB.NET

```
Private Sub OleEvent(ByVal sender As System.Object, ByVal e As
AxEXSTATUSBARLib._IStatusBarEvents_OleEventEvent) Handles OleEvent
End Sub
```

VB6

```
Private Sub OleEvent(ByVal Panel As EXSTATUSBARLibCtl.IPanel,ByVal Ev As
EXSTATUSBARLibCtl.IOleEvent)
End Sub
```

VBA

```
Private Sub OleEvent(ByVal Panel As Object,ByVal Ev As Object)
End Sub
```

VFP

```
LPARAMETERS Panel,Ev
```

Xbas…

```
PROCEDURE OnOleEvent(oStatusBar,Panel,Ev)
RETURN
```

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

Java…

```
<SCRIPT EVENT="OleEvent(Panel,Ev)" LANGUAGE="JScript">
</SCRIPT>
```

VBSc…

```
<SCRIPT LANGUAGE="VBScript">
Function OleEvent(Panel,Ev)
End Function
```

```
</SCRIPT>
```

Visual Data...

```
Procedure OnComOleEvent Variant llPanel Variant llEv
    Forward Send OnComOleEvent llPanel llEv
End_Procedure
```

Visual Objects

```
METHOD OCX_OleEvent(Panel,Ev) CLASS MainDialog
RETURN NIL
```

X++

```
void onEvent_OleEvent(COM _Panel,COM _Ev)
{
}
```

XBasic

```
function OleEvent as v (Panel as OLE::Exontrol.StatusBar.1::IPanel,Ev as
OLE::Exontrol.StatusBar.1::IOleEvent)
end function
```

dBASE

```
function nativeObject_OleEvent(Panel,Ev)
return
```

The following VB sample enumerates the arguments of an OLE event when OLEEvent is fired.

```
Private Sub StatusBar1_OleEvent(ByVal Panel As EXSTATUSBARLibCtl.IPanel, ByVal Ev As
EXSTATUSBARLibCtl.IOleEvent)
    Debug.Print Ev.ToString()
End Sub
```

The following VC sample displays the events that an ActiveX control is firing while it is hosted by an item:

```
#import "C:\\WINNT\\SYSTEM32\\ExStatusBar.dll"
using namespace EXSTATUSBARLib;

void OnOleEventStatusbar1(LPDISPATCH Panel, LPDISPATCH Ev)
{
    EXSTATUSBARLib::IOleEventPtr spEvent( Ev );
    OutputDebugString( spEvent->ToString );
```

```
}
```

The #import clause is required to get the wrapper classes for IOleEvent and IOleEventParam objects, that are not defined by the MFC class wizard. The same #import statement defines the EXSTATUSBARLib namespace that include all objects and types of the control's TypeLibrary. In case your exstatusbar.dll library is located to another place than the system folder or well known path, the path to the library should be provided, in order to let the VC finds the type library.

The following VB.NET sample displays the events that an ActiveX control is firing while it is hosted by an item:

```
Private Sub AxStatusBar1_OleEvent(ByVal sender As System.Object, ByVal e As
AxEXSTATUSBARLib._IStatusBarEvents_OleEventEvent) Handles AxStatusBar1.OleEvent
    Debug.Print(e.ev.ToString)
End Sub
```

The following C# sample displays the events that an ActiveX control is firing while it is hosted by an item:

```
private void axStatusBar1_OleEvent(object sender,
AxEXSTATUSBARLib._IStatusBarEvents_OleEventEvent e)
{
    System.Diagnostics.Debug.Print(e.ev.ToString);
}
```

The following VFP sample displays the events that an ActiveX control fires when it is hosted by an item:

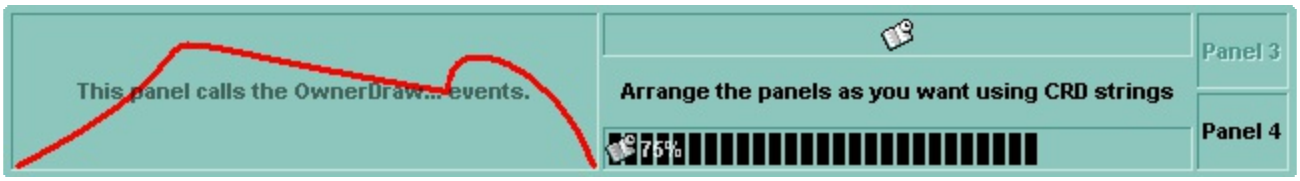
```
*** ActiveX Control Event ***
LPARAMETERS panel, ev

wait window nowait ev.ToString
```

event OwnerDrawEnd (Panel as Panel, hDC as Long)

Ends painting the owner draw panel.

Type	Description
Panel as Panel	A Panel object being painted
hDC as Long	A long expression that indicates the handle to the painting device context (HDC)

The control fires the OwnerDrawEnd event when painting the panel is done. *The* OwnerDrawEnd *event is fired after default painting is done*. The OwnerDrawStart event is fired when a panel requires to be painted. The OwnerDrawStart event is fired only for owner draw panels. Use the OwnerDraw property to specify which panel is owner draw and which panel is not. You can use the OwnerDrawStart event to avoid painting any panel using the DefaultPainting parameter. Use the OwnerDrawStart event to perform painting panel before default implementation is called. For instance, if the owner panel pains a transparent or lucent skin, the OwnerDrawStart event lets you paint the panel before putting the default skin. **The rectangle that should be painted in the device context can be retrieved using the GetClipBox API function**.



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

C#
```
private void OwnerDrawEnd(object sender,exontrol.EXSTATUSBARLib.Panel
Panel,int hDC)
{
}
```

VB
```
Private Sub OwnerDrawEnd(ByVal sender As System.Object,ByVal Panel As
exontrol.EXSTATUSBARLib.Panel,ByVal hDC As Integer) Handles OwnerDrawEnd
End Sub
```

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

C#
```
private void OwnerDrawEnd(object sender,
AxEXSTATUSBARLib._IStatusBarEvents_OwnerDrawEndEvent e)
```

```
{
}
```

C++

```
void OnOwnerDrawEnd(LPDISPATCH Panel,long hDC)
{
}
```

C++ Builder

```
void __fastcall OwnerDrawEnd(TObject *Sender,Exstatusbarlib_tlb::IPanel
*Panel,long hDC)
{
}
```

Delphi

```
procedure OwnerDrawEnd(ASender: TObject; Panel : IPanel;hDC : Integer);
begin
end;
```

Delphi 8 (.NET only)

```
procedure OwnerDrawEnd(sender: System.Object; e:
AxEXSTATUSBARLib._IStatusBarEvents_OwnerDrawEndEvent);
begin
end;
```

Powe…

```
begin event OwnerDrawEnd(oleobject Panel,long hDC)
end event OwnerDrawEnd
```

VB.NET

```
Private Sub OwnerDrawEnd(ByVal sender As System.Object, ByVal e As
AxEXSTATUSBARLib._IStatusBarEvents_OwnerDrawEndEvent) Handles
OwnerDrawEnd
End Sub
```

VB6

```
Private Sub OwnerDrawEnd(ByVal Panel As EXSTATUSBARLibCtl.IPanel,ByVal hDC
As Long)
End Sub
```

VBA

```
Private Sub OwnerDrawEnd(ByVal Panel As Object,ByVal hDC As Long)
End Sub
```

VFP

```
LPARAMETERS Panel,hDC
```

Xbas…

```
PROCEDURE OnOwnerDrawEnd(oStatusBar,Panel,hDC)
RETURN
```

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

Java…

```
<SCRIPT EVENT="OwnerDrawEnd(Panel,hDC)" LANGUAGE="JScript">
</SCRIPT>
```

VBSc…

```
<SCRIPT LANGUAGE="VBScript">
Function OwnerDrawEnd(Panel,hDC)
End Function
</SCRIPT>
```

Visual Data…

```
Procedure OnComOwnerDrawEnd Variant llPanel Integer llhDC
   Forward Send OnComOwnerDrawEnd llPanel llhDC
End_Procedure
```

Visual Objects

```
METHOD OCX_OwnerDrawEnd(Panel,hDC) CLASS MainDialog
RETURN NIL
```

X++

```
void onEvent_OwnerDrawEnd(COM _Panel,int _hDC)
{
}
```

XBasic

```
function OwnerDrawEnd as v (Panel as OLE::Exontrol.StatusBar.1::IPanel,hDC as N)
end function
```

dBASE

```
function nativeObject_OwnerDrawEnd(Panel,hDC)
return
```

The following VB sample displays a curve in the owner draw panel:

```
Private Type RECT
    Left As Long
    Top As Long
    Right As Long
    Bottom As Long
```

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

Private Declare Function GetClipBox Lib "gdi32" (ByVal hdc As Long, lpRect As RECT) As Long
Private Declare Function PolyBezier Lib "gdi32" (ByVal hdc As Long, lppt As POINTAPI, ByVal cPoints As Long) As Long
Private Declare Function SelectObject Lib "gdi32" (ByVal hdc As Long, ByVal hObject As Long) As Long
Private Declare Function CreatePen Lib "gdi32" (ByVal nPenStyle As Long, ByVal nWidth As Long, ByVal crColor As Long) As Long
Private Declare Function DeleteObject Lib "gdi32" (ByVal hObject As Long) As Long

Private Sub StatusBar1_OwnerDrawStart(ByVal Panel As EXSTATUSBARLibCtl.IPanel, ByVal hdc As Long, DefaultPainting As Boolean)
  Dim r As RECT
  GetClipBox hdc, r
  hPen = CreatePen(0, 3, RGB(255, 0, 0))
  hOPen = SelectObject(hdc, hPen)
  r.Left = r.Left + 4
  r.Right = r.Right - 4
  r.Top = r.Top + 4
  r.Bottom = r.Bottom - 4
  Dim p(7) As POINTAPI
  p(0).x = r.Left
  p(0).y = r.Bottom
  p(1).x = (r.Left + r.Right) / 2
  p(1).y = r.Top
  p(2).x = r.Left
  p(2).y = r.Top
  p(3).x = 2 * (r.Left + r.Right) / 3
  p(3).y = (r.Bottom + r.Top) / 2
  p(4).x = 2 * (r.Left + r.Right) / 3
  p(4).y = r.Top
```

```
    p(5).x = 4 * (r.Left + r.Right) / 5
    p(5).y = (r.Bottom + r.Top) / 3
    p(6).x = r.Right
    p(6).y = r.Bottom
    PolyBezier hdc, p(0), 7
    SelectObject hdc, hOPen
    DeleteObject hOPen
End Sub
```

The following C++ sample displays a curve in the owner draw panel:

```
void OnOwnerDrawStartStatusbar1(LPDISPATCH Panel, long hDC, BOOL FAR*
DefaultPainting)
{
    HDC h = (HDC)hDC;
    RECT r = {0};

    GetClipBox( h, &r );
    CPanel panel(Panel);
    panel.m_bAutoRelease = FALSE;
    POINT p[7] = {(0,0)};
    HPEN hPen = CreatePen( PS_SOLID, 3, RGB(255,0,0) );
    HPEN hOPen = (HPEN)::SelectObject( h, hPen );
    InflateRect( &r, -4, -4 );

    p[0].x = r.left;
    p[0].y = r.bottom;
    p[1].x = (r.left + r.right) / 2;
    p[1].y = r.top;
    p[2].x = r.left;
    p[2].y = r.top;
    p[3].x = 2 * (r.left + r.right) / 3;
    p[3].y = (r.bottom + r.top) / 2;
    p[4].x = 2 * (r.left + r.right) / 3;
    p[4].y = r.top;
    p[5].x = 4 * (r.left + r.right) / 5;
    p[5].y = (r.bottom + r.top) / 3;
```

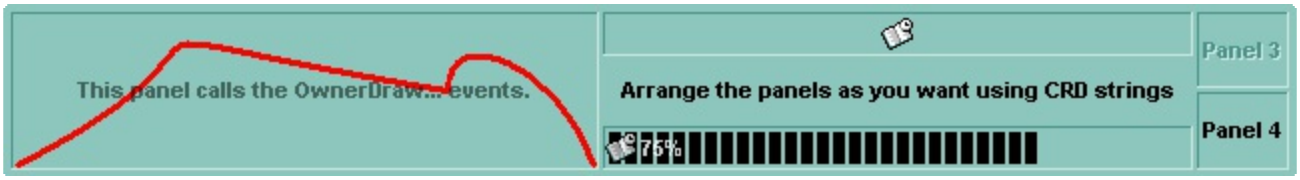
```
    p[6].x = r.right;
    p[6].y = r.bottom;
    PolyBezier( h, p, 7 );

    ::SelectObject( h, hOPen );
    DeleteObject( hPen );
}
```

event OwnerDrawStart (Panel as Panel, hDC as Long, DefaultPainting as Boolean)

Starts painting the owner draw panel.

Type	Description
Panel as Panel	A Panel object being painted
hDC as Long	A long expression that indicates the handle to the painting device context (HDC)
DefaultPainting as Boolean	A Boolen expression that indicates whether the default painting should be performed or not. If the DefaultPainting parameter is True, the control paints the part as default, else the panel is not painted by the control so the user should draw the panel.

The OwnerDrawStart event is fired when a panel requires to be painted. *The OwnerDrawStart event is fired before erasing and painting the panel*. The OwnerDrawStart event is fired only for owner draw panels. Use the OwnerDraw property to specify which panel is owner draw and which panel is not. You can use the OwnerDrawStart event to avoid painting any panel using the DefaultPainting parameter. The control fires the OwnerDrawEnd event when painting the panel is done. Use the OwnerDrawStart event to perform painting panel before default implementation is called. For instance, if the owner panel pains a transparent or lucent skin, the OwnerDrawStart event lets you paint the panel before putting the default skin. **The rectangle that should be painted in the device context can be retrieved using the GetClipBox API function**.



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

C#

```
private void OwnerDrawStart(object sender,exontrol.EXSTATUSBARLib.Panel Panel,int hDC,ref bool DefaultPainting)
{
}
```

VB

```
Private Sub OwnerDrawStart(ByVal sender As System.Object,ByVal Panel As exontrol.EXSTATUSBARLib.Panel,ByVal hDC As Integer,ByRef DefaultPainting As Boolean) Handles OwnerDrawStart
```

```
End Sub
```

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

C#

```
private void OwnerDrawStart(object sender,
AxEXSTATUSBARLib._IStatusBarEvents_OwnerDrawStartEvent e)
{
}
```

C++

```
void OnOwnerDrawStart(LPDISPATCH Panel,long hDC,BOOL FAR* DefaultPainting)
{
}
```

C++ Builder

```
void __fastcall OwnerDrawStart(TObject *Sender,Exstatusbarlib_tlb::IPanel
*Panel,long hDC,VARIANT_BOOL * DefaultPainting)
{
}
```

Delphi

```
procedure OwnerDrawStart(ASender: TObject; Panel : IPanel;hDC : Integer;var
DefaultPainting : WordBool);
begin
end;
```

Delphi 8 (.NET only)

```
procedure OwnerDrawStart(sender: System.Object; e:
AxEXSTATUSBARLib._IStatusBarEvents_OwnerDrawStartEvent);
begin
end;
```

Powe…

```
begin event OwnerDrawStart(oleobject Panel,long hDC,boolean DefaultPainting)
end event OwnerDrawStart
```

VB.NET

```
Private Sub OwnerDrawStart(ByVal sender As System.Object, ByVal e As
AxEXSTATUSBARLib._IStatusBarEvents_OwnerDrawStartEvent) Handles
OwnerDrawStart
End Sub
```

VB6

```
Private Sub OwnerDrawStart(ByVal Panel As EXSTATUSBARLibCtl.IPanel,ByVal hDC
As Long,DefaultPainting As Boolean)
```

```
End Sub
```

VBA

```
Private Sub OwnerDrawStart(ByVal Panel As Object,ByVal hDC As
Long,DefaultPainting As Boolean)
End Sub
```

VFP

```
LPARAMETERS Panel,hDC,DefaultPainting
```

Xbas…

```
PROCEDURE OnOwnerDrawStart(oStatusBar,Panel,hDC,DefaultPainting)
RETURN
```

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

Java…

```
<SCRIPT EVENT="OwnerDrawStart(Panel,hDC,DefaultPainting)"
LANGUAGE="JScript">
</SCRIPT>
```

VBSc…

```
<SCRIPT LANGUAGE="VBScript">
Function OwnerDrawStart(Panel,hDC,DefaultPainting)
End Function
</SCRIPT>
```

Visual Data…

```
Procedure OnComOwnerDrawStart Variant llPanel Integer llhDC Boolean
llDefaultPainting
   Forward Send OnComOwnerDrawStart llPanel llhDC llDefaultPainting
End_Procedure
```

Visual Objects

```
METHOD OCX_OwnerDrawStart(Panel,hDC,DefaultPainting) CLASS MainDialog
RETURN NIL
```

X++

```
void onEvent_OwnerDrawStart(COM _Panel,int _hDC,COMVariant /*bool*/
_DefaultPainting)
{
}
```

XBasic

```
function OwnerDrawStart as v (Panel as OLE::Exontrol.StatusBar.1::IPanel,hDC as
N,DefaultPainting as L)
```

```
end function
```

dBASE

```
function nativeObject_OwnerDrawStart(Panel,hDC,DefaultPainting)
return
```

The following VB sample displays a curve in the owner draw panel:

```
Private Type RECT
    Left As Long
    Top As Long
    Right As Long
    Bottom As Long
End Type
Private Type POINTAPI
    x As Long
    y As Long
End Type

Private Declare Function GetClipBox Lib "gdi32" (ByVal hdc As Long, lpRect As RECT) As Long
Private Declare Function PolyBezier Lib "gdi32" (ByVal hdc As Long, lppt As POINTAPI, ByVal cPoints As Long) As Long
Private Declare Function SelectObject Lib "gdi32" (ByVal hdc As Long, ByVal hObject As Long) As Long
Private Declare Function CreatePen Lib "gdi32" (ByVal nPenStyle As Long, ByVal nWidth As Long, ByVal crColor As Long) As Long
Private Declare Function DeleteObject Lib "gdi32" (ByVal hObject As Long) As Long

Private Sub StatusBar1_OwnerDrawStart(ByVal Panel As EXSTATUSBARLibCtl.IPanel, ByVal hdc As Long, DefaultPainting As Boolean)
  Dim r As RECT
  GetClipBox hdc, r
  hPen = CreatePen(0, 3, RGB(255, 0, 0))
  hOPen = SelectObject(hdc, hPen)
  r.Left = r.Left + 4
  r.Right = r.Right - 4
  r.Top = r.Top + 4
```

```
    r.Bottom = r.Bottom - 4
    Dim p(7) As POINTAPI
    p(0).x = r.Left
    p(0).y = r.Bottom
    p(1).x = (r.Left + r.Right) / 2
    p(1).y = r.Top
    p(2).x = r.Left
    p(2).y = r.Top
    p(3).x = 2 * (r.Left + r.Right) / 3
    p(3).y = (r.Bottom + r.Top) / 2
    p(4).x = 2 * (r.Left + r.Right) / 3
    p(4).y = r.Top
    p(5).x = 4 * (r.Left + r.Right) / 5
    p(5).y = (r.Bottom + r.Top) / 3
    p(6).x = r.Right
    p(6).y = r.Bottom
    PolyBezier hdc, p(0), 7
    SelectObject hdc, hOPen
    DeleteObject hOPen
End Sub
```

The following C++ sample displays a curve in the owner draw panel:

```
void OnOwnerDrawStartStatusbar1(LPDISPATCH Panel, long hDC, BOOL FAR*
DefaultPainting)
{
    HDC h = (HDC)hDC;
    RECT r = {0};

    GetClipBox( h, &r );
    CPanel panel(Panel);
    panel.m_bAutoRelease = FALSE;
    POINT p[7] = {(0,0)};
    HPEN hPen = CreatePen( PS_SOLID, 3, RGB(255,0,0) );
    HPEN hOPen = (HPEN)::SelectObject( h, hPen );
    InflateRect( &r, -4, -4 );
```

```
    p[0].x = r.left;
    p[0].y = r.bottom;
    p[1].x = (r.left + r.right) / 2;
    p[1].y = r.top;
    p[2].x = r.left;
    p[2].y = r.top;
    p[3].x = 2 * (r.left + r.right) / 3;
    p[3].y = (r.bottom + r.top) / 2;
    p[4].x = 2 * (r.left + r.right) / 3;
    p[4].y = r.top;
    p[5].x = 4 * (r.left + r.right) / 5;
    p[5].y = (r.bottom + r.top) / 3;
    p[6].x = r.right;
    p[6].y = r.bottom;
    PolyBezier( h, p, 7 );

    ::SelectObject( h, hOPen );
    DeleteObject( hPen );
}
```

event PercentChange (Panel as Panel)

Occurs when the Percent property is changed.

Type	Description
Panel as Panel	A Panel object where the Percent property has been changed.

Use the PercentChange event to notify your application when the Percent property is changed. The Percent property specifies the percent of the progress bar being displayed in the panel. Use the PercentChange event to change the panel's caption according to the percent value. The Text property specifies the caption of the panel.

The following VB sample change the panel's caption according to the Percent value:

```
Private Sub StatusBar1_PercentChange(ByVal Panel As EXSTATUSBARLibCtl.IPanel)
   Panel.Text = Panel.Percent + "%"
End Sub
```

The following VB.NET sample change the panel's caption according to the Percent value:

```
Private Sub AxStatusBar1_PercentChange(ByVal sender As System.Object, ByVal e As
AxEXSTATUSBARLib._IStatusBarEvents_PercentChangeEvent) Handles
AxStatusBar1.PercentChange
   With e.panel
      .Text = .Percent.ToString() + "%"
   End With
End Sub
```

The following C# sample change the panel's caption according to the Percent value:

```
private void axStatusBar1_PercentChange(object sender,
AxEXSTATUSBARLib._IStatusBarEvents_PercentChangeEvent e)
{
   e.panel.Text = e.panel.Percent.ToString() + "%";
}
```

The following C++ sample change the panel's caption according to the Percent value:

```
void OnPercentChangeStatusbar1(LPDISPATCH Panel)
{
```

```
    CPanel panel(Panel);
    panel.m_bAutoRelease = FALSE;
    CString strPercent;
    strPercent.Format( "%i%%", panel.GetPercent() );
    panel.SetText( strPercent );
}
```

The following VFP sample change the panel's caption according to the Percent value:

```
*** ActiveX Control Event ***
LPARAMETERS panel

with panel
   .Text = LTrim(RTrim(.Percent)) + "%"
endwith
```

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

C#

```
private void PercentChange(object sender,exontrol.EXSTATUSBARLib.Panel Panel)
{
}
```

VB

```
Private Sub PercentChange(ByVal sender As System.Object,ByVal Panel As
exontrol.EXSTATUSBARLib.Panel) Handles PercentChange
End Sub
```

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

C#

```
private void PercentChange(object sender,
AxEXSTATUSBARLib._IStatusBarEvents_PercentChangeEvent e)
{
}
```

C++

```
void OnPercentChange(LPDISPATCH Panel)
{
}
```

C++ Builder

```
void __fastcall PercentChange(TObject *Sender,Exstatusbarlib_tlb::IPanel *Panel)
{
```

```
}
```

Delphi

```
procedure PercentChange(ASender: TObject; Panel : IPanel);
begin
end;
```

Delphi 8 (.NET only)

```
procedure PercentChange(sender: System.Object; e:
AxEXSTATUSBARLib._IStatusBarEvents_PercentChangeEvent);
begin
end;
```

Powe…

```
begin event PercentChange(oleobject Panel)
end event PercentChange
```

VB.NET

```
Private Sub PercentChange(ByVal sender As System.Object, ByVal e As
AxEXSTATUSBARLib._IStatusBarEvents_PercentChangeEvent) Handles
PercentChange
End Sub
```

VB6

```
Private Sub PercentChange(ByVal Panel As EXSTATUSBARLibCtl.IPanel)
End Sub
```

VBA

```
Private Sub PercentChange(ByVal Panel As Object)
End Sub
```

VFP

```
LPARAMETERS Panel
```

Xbas…

```
PROCEDURE OnPercentChange(oStatusBar,Panel)
RETURN
```

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

Java…

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

VBSc…

```
<SCRIPT LANGUAGE="VBScript">
Function PercentChange(Panel)
```

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

Visual Data…

```
Procedure OnComPercentChange Variant llPanel
   Forward Send OnComPercentChange llPanel
End_Procedure
```

Visual Objects

```
METHOD OCX_PercentChange(Panel) CLASS MainDialog
RETURN NIL
```

X++

```
void onEvent_PercentChange(COM _Panel)
{
}
```

XBasic

```
function PercentChange as v (Panel as OLE::Exontrol.StatusBar.1::IPanel)
end function
```

dBASE

```
function nativeObject_PercentChange(Panel)
return
```