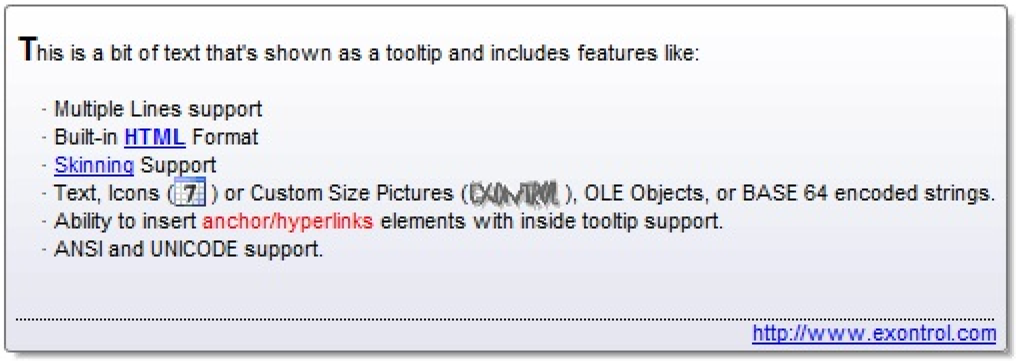
ExToolTip

The Exontrol's ExToolTip provides featured tooltips for your application. The tooltip is a common graphical user interface element. It is used in conjunction with a cursor, usually a mouse pointer. The user hovers the cursor over an item, without clicking it, and a small box appears with supplementary information regarding the item being hovered over. Adding the component to your projects is very easy, and requires only a few lines of code.

The features of the component include:

- Multi-lines support
- Built-in **HTML** format
- **Text Decorations** support, like gradient, outlined characters, shadow, and so on
- **Skinning** support, including round corners
- Customizable **Fonts** and **Colors**
- Text, Icons, or Custom Size **Pictures**
- PNG, TIFF, EXIF or WMF image format support
- Ability to insert **hyperlinks** with inside tooltip support
- Transparency support
- Configurable Timing



This is a bit of text that's shown as a tooltip and includes features like:

- Multiple Lines support
- Built-in HTML Format
- Skinning Support
- Text, Icons (7) or Custom Size Pictures (), OLE Objects, or BASE 64 encoded strings.
- Ability to insert anchor/hyperlinks elements with inside tooltip support.
- ANSI and UNICODE support.

http://www.exontrol.com

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

How to start?

The following steps show you progressively how to start programming the Exontrol's ExToolTip component:

- In **VB**
  - Add a reference to the ExToolTip 1.0 Control Library, using the Project\Reference
  - Add declaration for the component, like *Private WithEvents t As EXTOOLTIPLib.ToolTip*, if you plan to use the events of the control, else you can use *Dim t as New EXTOOLTIPLib.ToolTip*, to instantiate the control
  - Instantiate the component when the form is loading, as follows: *Set t = New EXTOOLTIPLib.Top*, and so your form may use the newly created tooltip
  - Specify the tooltip's appearance, icons or pictures using common properties like: Appearance, Images, HTMLPicture and so on
  - Call the *ShowToolTip* method when the mouse is moving.

```
Private Sub Form_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
    t.ShowToolTip "This is a bit of text that's shown when the cursor hovers the form"
End Sub
```

- In **VB.NET**
  - Add a reference to the ExToolTip 1.0 Control Library, using the Project\Add Reference\COM
  - Add declaration for the component, like *Private WithEvents t As EXTOOLTIPLib.ToolTip*, if you plan to use the events of the control, else you can use Dim t as New EXTOOLTIPLib.ToolTip, to instantiate the control
  - Instantiate the component when the form is loading, as follows: *Set t = New EXTOOLTIPLib.Top*, and so your form may use the newly created tooltip
  - Specify the tooltip's appearance, icons or pictures using common properties like: Appearance, Images, HTMLPicture and so on
  - Call the *ShowToolTip* method when the mouse is moving.

```
Private Sub Form1_MouseMove(ByVal sender As Object, ByVal e As System.Windows.Forms.MouseEventArgs) Handles Me.MouseMove
    t.ShowToolTip("This is a bit of text that's shown when the cursor hovers the form")
```

```
End Sub
```

- In **C#**
  - Add a reference to the ExToolTip 1.0 Control Library, using the Project\Add Reference\COM
  - Instantiate the component when the form is loading, as follows: *t = new EXTOOLTIPLib.ToolTip()*, and so your form may use the newly created tooltip
  - Add the handler for the AnchorClick event, in case you plan to use it as follows:

```
t.AnchorClick += new
EXTOOLTIPLib._IToolTipEvents_AnchorClickEventHandler(t_AnchorClick);
```

```
void t_AnchorClick(string AnchorID, string Options)
{
   System.Diagnostics.Debug.WriteLine("AnchorClick event");
}
```

  - Specify the tooltip's appearance, icons or pictures using common properties like: Appearance, Images, HTMLPicture and so on
  - Call the *ShowToolTip* method when the mouse is moving.

```
private void Form1_MouseMove(object sender, MouseEventArgs e)
{
   t.ShowToolTip("This is a bit of text that's shown when the cursor hovers the
form", "", "", "", "" );
}
```

- In **C++**
  - Import the Exontrol's ExToolTip type library using the *#import "c:/winnt/system32/extooltip.dll"*, so the EXTOOLTIPLib namespace is defined
  - Add a member of EXTOOLTIPLib::IToolTipPtr type to your dialog, as *EXTOOLTIPLib::IToolTipPtr m_spToolTip*
  - Instantiate the m_spToolTip member as follows:

```
CoInitialize( NULL );
if ( SUCCEEDED( CoCreateInstance( __uuidof(EXTOOLTIPLib::ToolTip), NULL,
CLSCTX_ALL, __uuidof(EXTOOLTIPLib::IToolTip), (LPVOID*)&m_spToolTip ) ) )
{
}
```

- Use the Notifier property to assign the handle of the dialog that receives notifications from the tooltip, if we plan to use the control's events.
- Specify the tooltip's appearance, icons or pictures using common properties like: Appearance, Images, HTMLPicture and so on
- Handle the WM_MOUSEMOVE message of the dialog, or add a handler to the PreTranslateMessage function to call the *ShowToolTip* method:

```
BOOL PreTranslateMessage(MSG* pMsg)
{
    /*
        Handles the WM_MOUSEMOVE message during the PreTranslateMessage so it can show the tooltip even if we move the mouse over the inside controls too.
        As the WM_MOUSEMOVE message is not sent to dialog, if the cursor hovers the inside windows...
    */
    if ( pMsg->message == WM_MOUSEMOVE )
    {
        if ( m_spToolTip != NULL )
            m_spToolTip->ShowToolTip( COleVariant( "This is a bit of text that's shown when the cursor hovers the form" ), vtMissing, vtMissing, vtMissing );
    }
    return CDialog::PreTranslateMessage(pMsg);
}
```

- In **VFP**
  - Create the Exontrol's ExToolTip object as follows:

```
public t as Object
t = CreateObject("Exontrol.ToolTip")
```

  - Specify the tooltip's appearance, icons or pictures using common properties like: Appearance, Images, HTMLPicture and so on
  - Call the *ShowToolTip* method during the Form's MouseMove event as follows:

```
LPARAMETERS nButton, nShift, nXCoord, nYCoord

with t
    .ShowToolTip("This is a bit of text that's shown when the cursor hovers the form")
```

```
endwith
```

- **Web**
  - Add a member to the page, as *Dim t*
  - Use the *window_onload* event to create the Exontrol's ExToolTip object as follows:

```
Sub window_onload
    set t = CreateObject("Exontrol.ToolTip")
End Sub
```

  - Specify the tooltip's appearance, icons or pictures using common properties like: Appearance, Images, HTMLPicture and so on
  - Call the *ShowToolTip* method during the *document_onmousemove* event as follows:

```
Sub document_onmousemove
    t.ShowToolTip "This is a bit of text that's shown when the cursor hovers the form"
End Sub
```

Send comments on this topic.

constants AlignmentEnum

The AlignmenEnum type specifies the alignment of the tooltip relative to the showing position. Use the Alignment parameter of the ShowToolTip method, to specify the alignment of the tooltip. The AlignmentEnum type supports the following values:

Name	Value	Description
exTopLeft	0	Aligns the object to the top/left corner.
exTopRight	1	Aligns the object to the top/right corner.
exBottomLeft	2	Aligns the object to the bottom/left corner.
exBottomRight	3	Aligns the object to the bottom/right corner.
exCenter	16	Aligns the object in the center.
exCenterLeft	17	Aligns the object in the center, to the left side.
exCenterRight	18	Aligns the object in the center, to the right side.
exCenterTop	19	Aligns the object in the center, to the top side.
exCenterBottom	19	Aligns the object in the center, to the bottom side.

constants AppearanceEnum

The AppearanceEnum type specifies the predefined types of borders for the tooltip. Use the Appearance property to change the tooltip's appearance. The AppearanceEnum type supports the following predefined values:

Name	Value	Description
exNone	0	No border
exFlat	1	Flat border
exSunken	2	Sunken border
exRaised	3	Raised border
exShadow	16	Shadow border

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)

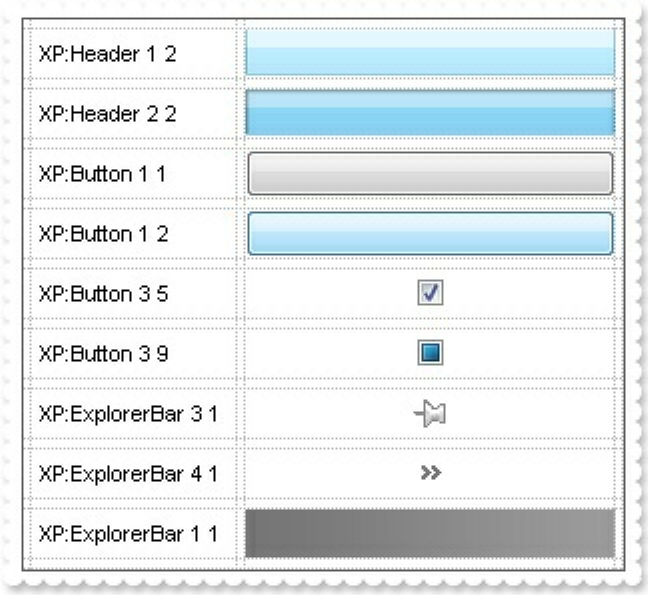
Adds or replaces a skin object to the control.

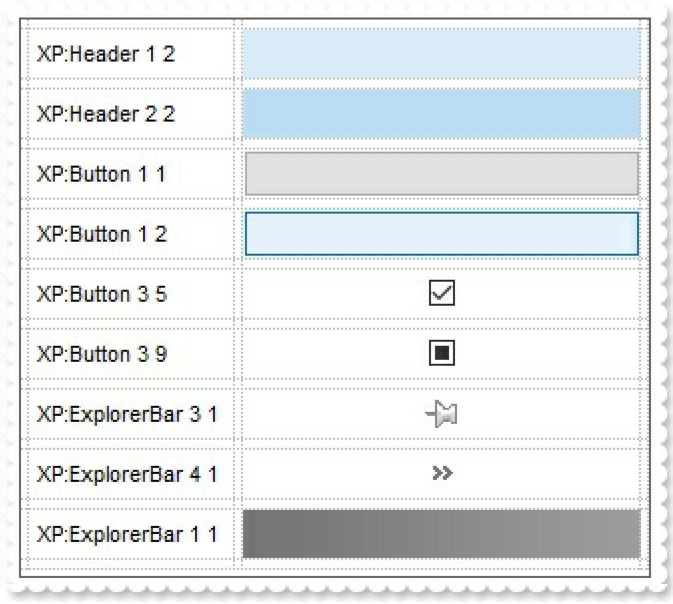
Type	Description
ID as Long	A Long expression that indicates the index of the skin being added or replaced. The value must be between 1 and 126, so Appearance collection should holds no more than 126 elements.
	The Skin parameter of the Add method can a STRING as explained bellow, a BYTE[] / safe arrays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

the end of this document. For instance the "XP:Header 1 2" indicates the part 1 of the Header class in the state 2, in the current Windows XP theme.

Skin as Variant

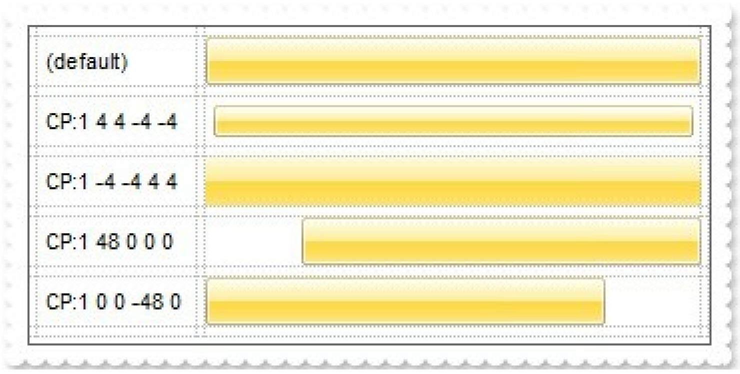
The following screen shots show a few Windows XP Theme Elements, running on Windows Vista and Windows 10, using the XP options:





- A copy of another skin with different coordinates (position, size), if the Skin parameter starts with "**CP**:". Use this option, to display the EBN, using different coordinates (position, size). By default, the EBN skin object is rendered on the part's client area. Using this option, you can display the same EBN, on a different position / size. In this case, the syntax of the Skin parameter is: "CP:ID Left Top Right Bottom" where the ID is the identifier of the EBN to be used (it is a number that specifies the ID parameter of the Add method), Left, Top, Right and Bottom parameters/numbers specifies the relative position to the part's client area, where the EBN should be rendered. The Left, Top, Right and Bottom parameters are numbers (negative, zero or positive values, with no decimal), that can be followed by the D character which indicates the value according to the current DPI settings. For instance, "CP:1 -2 -2 2 2", uses the EBN with the identifier 1, and displays it on a 2-pixels wider rectangle no matter of the DPI settings, while "CP:1 -2D -2D 2D 2D" displays it on a 2-pixels wider rectangle if DPI settings is 100%, and on on a 3-pixels wider rectangle if DPI settings is 150%.

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

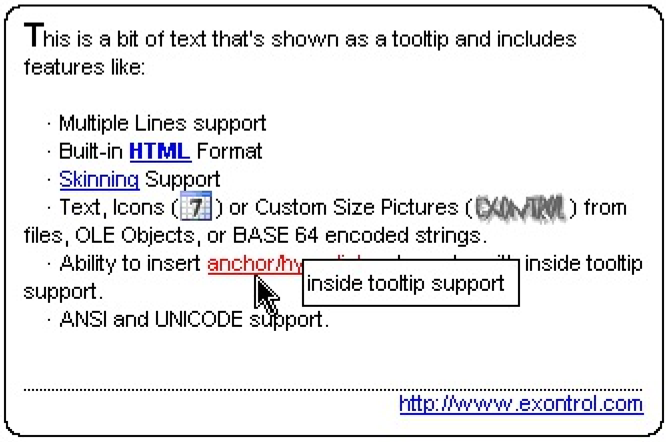


Return	Description
Boolean	A Boolean expression that indicates whether the new skin was added or replaced.

Use the Add method to add or replace skins to the control. The skin method, in it's simplest form, uses a single graphic file (*.ebn) assigned to a part of the control, 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 Appearance property to change the visual appearance of the borders of the tooltip. Use the BackColor property to specify the control's background color. Use the ForeColor property to specify the control's foreground color.

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 Appearance = &H2000000 indicates that the second skin object defines the border of the tooltip.


This is a bit of text that's shown as a tooltip and includes features like:

· Multiple Lines support
· Built-in HTML Format
· Skinning Support
· Text, Icons (7) or Custom Size Pictures ( ) from files, OLE Objects, or BASE 64 encoded strings.
· Ability to insert anchor/hy inside tooltip support.
· ANSI and UNICODE support.

inside tooltip support

http://www.exontrol.com

The following VB sample changes the border of the tooltip, using an EBN file:

```
Private Sub Form_Load()
   Set t = New EXTOOLTIPLib.ToolTip
   With t
      .VisualAppearance.Add &H12, "c:\temp\winword.ebn"
      t.Appearance = &H12000000
      t.BackColor = RGB(255, 255, 255)
   End With
End Sub
```

The following VB.NET sample changes the border of the tooltip, using an EBN file:

```
Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles MyBase.Load
   t = New EXTOOLTIPLib.ToolTip
   t.VisualAppearance.Add(&H12, "c:\temp\winword.ebn")
   t.Appearance = &H12000000
   t.BackColor = ToUInt32(Color.White)
End Sub
```

where the ToUInt32 function is defined like follows:

```
Shared Function ToUInt32(ByVal c As Color) As UInt32
    Dim i As Long
    i = c.R
    i = i + 256 * c.G
    i = i + 256 * 256 * c.B
    ToUInt32 = Convert.ToUInt32(i)
End Function
```

The following C# sample changes the border of the tooltip, using an EBN file:

```
private void Form1_Load(object sender, EventArgs e)
{
    t = new EXTOOLTIPLib.ToolTip();
    t.VisualAppearance.Add(0x12, "c:\\temp\\winword.ebn");
    t.Appearance = (EXTOOLTIPLib.AppearanceEnum)0x12000000;
    t.BackColor = ToUInt32(Color.White);
}
```

where the ToUInt32 function is defined like follows:

```
private UInt32 ToUInt32(Color c)
{
    long i;
    i = c.R;
    i = i + 256 * c.G;
    i = i + 256 * 256 * c.B;
    return Convert.ToUInt32(i);
}
```

The following C++ sample changes the border of the tooltip, using an EBN file:

```
void initToolTip()
{
    CoInitialize( NULL );
    if ( SUCCEEDED( CoCreateInstance( __uuidof(EXTOOLTIPLib::ToolTip), NULL, CLSCTX_ALL,
__uuidof(EXTOOLTIPLib::IToolTip), (LPVOID*)&m_spToolTip ) ) )
    {
        m_spToolTip->VisualAppearance->Add( 0x12, COleVariant( "c:\\temp\\winword.ebn"
) );
        m_spToolTip->Appearance = (EXTOOLTIPLib::AppearanceEnum)0x12000000;
        m_spToolTip->BackColor = RGB(255,255,255);
    }
}
```

The following VFP sample changes the border of the tooltip, using an EBN file:

```
public t as Object
t = CreateObject("Exontrol.ToolTip")

with t
    .VisualAppearance.Add(0x12,"c:\temp\winword.ebn")
    .Appearance = 0x12000000
    .BackColor = RGB(255,255,255)
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.

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.

ToolTip object

The Exontrol's ExToolTip provides featured tooltips for your application. The tooltip is a common graphical user interface element. It is used in conjunction with a cursor, usually a mouse pointer. The user hovers the cursor over an item, without clicking it, and a small box appears with supplementary information regarding the item being hovered over. Adding the component to your projects is very easy, and requires only a few lines of code. The ExToolTip components supports the following methods and properties:

Name	Description
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	Retrieves or sets a value that indicates the control's background color.
ExecuteTemplate	Executes a template and returns the result.
Font	Retrieves or sets the tooltip's font.
ForeColor	Retrieves or sets a value that indicates the control's foreground color.
FormatAnchor	Specifies the visual effect for anchor elements in the tooltip.
HideToolTip	Hides the tooltip.
HTMLPicture	Adds or replaces a picture in HTML text.
hWnd	Retrieves the control's window handle.
Images	Sets at runtime the control's image list. The Handle should be a handle to an Image List Control.
ImageSize	Retrieves or sets the size of icons the control displays..
Notifier	Retrieves or sets the window that receives notifications.
ReplaceIcon	Adds a new icon, replaces an icon or clears the control's image list.
ShowToolTip	Shows the tooltip.
Template	Specifies the control's template.
TemplateDef	Defines inside variables for the next Template/ExecuteTemplate call.
TemplatePut	Defines inside variables for the next Template/ExecuteTemplate call.

ToolTipDelay	Specifies the time in ms that passes before the ToolTip appears.
ToolTipHeight	Specifies a value that indicates the height of the ToolTip, in pixels.
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, in pixels.
Transparency	Specifies the tooltip's transparency.
TransparencyInside	Specifies the inside tooltip's transparency.
Version	Retrieves the control's version.
Visible	Specifies whether the tooltip is visible.
VisualAppearance	Retrieves the control's appearance.

property ToolTip.Appearance as AppearanceEnum

Retrieves or sets the control's appearance.

Type	Description
AppearanceEnum	An AppearanceEnum expression that indicates the tooltip'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 tooltip'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 text, icons or pictures of the tooltip are always shown in the tooltip's client area. The skin may contain transparent objects, and so you can define round corners. Use the eXButton's Skin builder to view or change this file***

Use the Appearance property to specify the tooltip's border. Use the BackColor property to specify the tooltip's background color. Use the ForeColor property to specify the tooltip's foreground color. Use the VisualAppearance property to access the tooltip's Appearance collection.

The following VB sample changes the border of the tooltip, using an EBN file:

```
Private Sub Form_Load()
   Set t = New EXTOOLTIPLib.ToolTip
   With t
      .VisualAppearance.Add &H12, "c:\temp\winword.ebn"
      t.Appearance = &H12000000
      t.BackColor = RGB(255, 255, 255)
   End With
End Sub
```

The following VB.NET sample changes the border of the tooltip, using an EBN file:

```
Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles MyBase.Load
   t = New EXTOOLTIPLib.ToolTip
   t.VisualAppearance.Add(&H12, "c:\temp\winword.ebn")
```

```
    t.Appearance = &H12000000
    t.BackColor = ToUInt32(Color.White)
End Sub
```

where the ToUInt32 function is defined like follows:

```
Shared Function ToUInt32(ByVal c As Color) As UInt32
    Dim i As Long
    i = c.R
    i = i + 256 * c.G
    i = i + 256 * 256 * c.B
    ToUInt32 = Convert.ToUInt32(i)
End Function
```

The following C# sample changes the border of the tooltip, using an EBN file:

```
private void Form1_Load(object sender, EventArgs e)
{
    t = new EXTOOLTIPLib.ToolTip();
    t.VisualAppearance.Add(0x12, "c:\\temp\\winword.ebn");
    t.Appearance = (EXTOOLTIPLib.AppearanceEnum)0x12000000;
    t.BackColor = ToUInt32(Color.White);
}
```

where the ToUInt32 function is defined like follows:

```
private UInt32 ToUInt32(Color c)
{
    long i;
    i = c.R;
    i = i + 256 * c.G;
    i = i + 256 * 256 * c.B;
    return Convert.ToUInt32(i);
}
```

The following C++ sample changes the border of the tooltip, using an EBN file:

```
void initToolTip()
{
    CoInitialize( NULL );
    if ( SUCCEEDED( CoCreateInstance( __uuidof(EXTOOLTIPLib::ToolTip), NULL, CLSCTX_ALL,
__uuidof(EXTOOLTIPLib::IToolTip), (LPVOID*)&m_spToolTip ) ) )
    {
        m_spToolTip->VisualAppearance->Add( 0x12, COleVariant( "c:\\temp\\winword.ebn"
) );
        m_spToolTip->Appearance = (EXTOOLTIPLib::AppearanceEnum)0x12000000;
        m_spToolTip->BackColor = RGB(255,255,255);
    }
}
```

The following VFP sample changes the border of the tooltip, using an EBN file:

```
public t as Object
t = CreateObject("Exontrol.ToolTip")

with t
    .VisualAppearance.Add(0x12,"c:\temp\winword.ebn")
    .Appearance = 0x12000000
    .BackColor = RGB(255,255,255)
endwith
```

method ToolTip.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 ToolTip1_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 ToolTip.BackColor as Color

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

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

By default, the BackColor property is 0. If the BackColor property is 0, the system defines the tooltip's background color. Use the BackColor property to specify the tooltip's background color. Use the Appearance property to specify the tooltip's border. Use the ForeColor property to specify the tooltip's foreground color. Use the <*bgcolor*> built-in HTML element to specify a background color for parts of the text in the tooltip.

The following VB sample changes the border of the tooltip, using an EBN file:

```
Private Sub Form_Load()
    Set t = New EXTOOLTIPLib.ToolTip
    With t
        .VisualAppearance.Add &H12, "c:\temp\winword.ebn"
        t.Appearance = &H12000000
        t.BackColor = RGB(255, 255, 255)
    End With
End Sub
```

The following VB.NET sample changes the border of the tooltip, using an EBN file:

```
Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles MyBase.Load
    t = New EXTOOLTIPLib.ToolTip
    t.VisualAppearance.Add(&H12, "c:\temp\winword.ebn")
    t.Appearance = &H12000000
    t.BackColor = ToUInt32(Color.White)
End Sub
```

where the ToUInt32 function is defined like follows:

```
Shared Function ToUInt32(ByVal c As Color) As UInt32
    Dim i As Long
    i = c.R
    i = i + 256 * c.G
    i = i + 256 * 256 * c.B
    ToUInt32 = Convert.ToUInt32(i)
End Function
```

The following C# sample changes the border of the tooltip, using an EBN file:

```
private void Form1_Load(object sender, EventArgs e)
{
    t = new EXTOOLTIPLib.ToolTip();
    t.VisualAppearance.Add(0x12, "c:\\temp\\winword.ebn");
    t.Appearance = (EXTOOLTIPLib.AppearanceEnum)0x12000000;
    t.BackColor = ToUInt32(Color.White);
}
```

where the ToUInt32 function is defined like follows:

```
private UInt32 ToUInt32(Color c)
{
    long i;
    i = c.R;
    i = i + 256 * c.G;
    i = i + 256 * 256 * c.B;
    return Convert.ToUInt32(i);
}
```

The following C++ sample changes the border of the tooltip, using an EBN file:

```
void initToolTip()
{
    CoInitialize( NULL );
    if ( SUCCEEDED( CoCreateInstance( __uuidof(EXTOOLTIPLib::ToolTip), NULL, CLSCTX_ALL,
__uuidof(EXTOOLTIPLib::IToolTip), (LPVOID*)&m_spToolTip ) ) )
    {
        m_spToolTip->VisualAppearance->Add( 0x12, COleVariant( "c:\\temp\\winword.ebn"
) );
        m_spToolTip->Appearance = (EXTOOLTIPLib::AppearanceEnum)0x12000000;
        m_spToolTip->BackColor = RGB(255,255,255);
    }
}
```

The following VFP sample changes the border of the tooltip, using an EBN file:

```
public t as Object
t = CreateObject("Exontrol.ToolTip")

with t
    .VisualAppearance.Add(0x12,"c:\temp\winword.ebn")
    .Appearance = 0x12000000
    .BackColor = RGB(255,255,255)
endwith
```

method ToolTip.ExecuteTemplate (Template as String)

Executes a template and returns the result.

Type	Description
Template as String	A Template string being executed

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

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

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

```
Debug.Print ToolTip1.ExecuteTemplate("BackColor")
```

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

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

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

The Template 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 ToolTip.Font as IFontDisp

Retrieves or sets the font for objects that match the criteria.

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

Use the Font property to change the control's font. Use the <*font*> built-in HTML element to specify a different font for parts of the text in the tooltip. Use the <b>, <i>, <s> or <u> built-in HTML element to change the font's attributes to show portions of text inside the tooltip. Use the ForeColor property to specify the tooltip's foreground color. Use the <*bgcolor*> built-in HTML element to specify a background color for parts of the text in the tooltip.

property ToolTip.ForeColor as Color

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

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

By default, the ForeColor property is 0. If the ForeColor property is 0, the system defines the tooltip's foreground color. Use the ForeColor property to specify the control's foreground color. Use the Appearance property to specify the tooltip's border. Use the BackColor property to specify the tooltip's background color. Use the <*fgcolor*> built-in HTML element to specify a foreground color for parts of the text in the tooltip.

property ToolTip.FormatAnchor(New as Boolean) as String

Specifies the visual effect for anchor elements in the tooltip.

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

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

method ToolTip.HideToolTip ()

Hides the tooltip.

Type	Description

Use the HideToolTip method to programmatically hide the tooltip. Use the ShowToolTip method to show the tooltip. Use the hWnd property to get the handle of the window that hosts the tooltip. Use the Visible property to specify whether the tooltip is visible or hidden.

property ToolTip.HTMLPicture(Key as String) as Variant

Adds or replaces a picture in HTML text.

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> built-in HTML elements. By default, the HTMLPicture collection is empty. Use the Images method to assign a list of icons to the tooltip. Use the RepalceIcon method to add, remove or clear icons in the control's images collection. 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).

property ToolTip.hWnd as Long

Retrieves the control's window handle.

Type	Description
Long	A long expression that indicates the control's 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 ToolTip.Images (Handle as Variant)

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

Type	Description
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)

Use the Images method to assign a list of icons to the tooltip. The ImageSize property defines the size (width/height) of the icons within the control's Images collection. Use the ReplaceIcon method to add, remove or clear icons in the control's images collection. The HTMLPicture property handles a collection of custom size picture being displayed in the HTML captions, using the <img> built-in HTML elements.

property ToolTip.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 ToolTip.Notifier as Long

Retrieves or sets the window that receives notifications.

Type	Description
Long	A long expression that specifies the handle of the window to receive the notifications

Reserved for internal use only.

method ToolTip.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 method to add, remove or clear icons in the control's images collection. Use the Images method to assign a list of icons to the tooltip. The ImageSize property defines the size (width/height) of the icons within the control's Images collection. The HTMLPicture property handles a collection of custom size picture being displayed in the HTML captions, using the <img> built-in HTML elements. Also, the ReplaceIcon property can clear the images collection.

The following VB sample adds a new icon to control's images list:

i = ExToolTip1.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 = ExToolTip1.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:

ExToolTip1.ReplaceIcon 0, i, i specifies the index of icon removed.

The following VB clears the control's icons collection:

ExToolTip1.ReplaceIcon 0, -1

method ToolTip.ShowToolTip (ToolTip as Variant, [Title as Variant], [Alignment as Variant], [X as Variant], [Y as Variant])

Shows the tooltip.

Type	Description
ToolTip as Variant	A String expression that indicates the description of the tooltip, that supports built-in HTML format like described bellow.
Title as Variant	If present, A String expression that indicates the title of the tooltip.
Alignment as Variant	A long expression that indicates the alignment of the tooltip relative to the position of the cursor. If missing, the tooltip is aligned to the left/top corder.
X as Variant	A single that specifies the current X location of the mouse pointer. The x values is always expressed in screen coordinates. If missing or -1, the current mouse X position is used. A string expression that indicates the offset to move the tooltip window relative to the cursor position.
Y as Variant	A single that specifies the current Y location of the mouse pointer. The y values is always expressed in screen coordinates. If missing or -1, the current mouse Y position is used. A string expression that indicates the offset to move the tooltip window relative to the cursor position.

Use the ShowToolTip method to display programmatically the tooltip. Use the HideToolTip method to hide the tooltip. *The ToolTipDelay property specifies the time in ms that passes before the ToolTip appears.* Use the ToolTipPopDelay property specifies the period in ms of time the ToolTip remains visible if the mouse pointer is stationary within a control. Use the Font property to change the tooltip's font. Use the Appearance property indicates the visual appearance of the borders of the tooltip. Use the BackColor property indicates the tooltip's background color. Use the ForeColor property indicates the tooltip's foreground color.

The **ToolTip** parameter supports the following HTML elements:

- **<b> ... </b>** displays the text in **bold**
- **<i> ... </i>** displays the text in *italics*
- **<u> ... </u>** underlines the text
- **<s> ... </s>** ~~Strike~~-through text
- **<a id;options> ... </a>** 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.

- **<font face;size> ... </font>** displays portions of text with a different font and/or different size. For instance, the "<font Tahoma;12>bit</font>" 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, "<font ;12>bit</font>" 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
- **<br>** 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 **&amp;** (&), **&lt;** (<), **&gt;** (>), **&qout;** (") and **&#number;** (the character with specified code), For instance, the &#8364; 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 <b>bold</b> in HTML caption you can use &lt;b&gt;bold&lt;/b&gt;

- **<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 <font face;size> to define a smaller or a larger font to be displayed. For instance: "Text with <font ;7><**off** 6>subscript" displays the text such as: Text with $_{\text{subscript}}$ The "Text with <font ;7><**off** -6>superscript" displays the text such as: Text with $^{\text{subscript}}$
- **<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 <font> 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 "<font ;18><**gra** FFFFFF;1;1>gradient-center</**gra**></font>" 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 <font> HTML tag can be used to define the height of the font. For instance the "<font ;31><**out** 000000><fgcolor=FFFFFF>outlined</fgcolor></**out**></font>" 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 <font> HTML tag can be used to define the height of the font. For instance the "<font ;31><**sha**>shadow</**sha**></font>" generates the following picture:

or "*<font ;31><**sha** 404040;5;0><fgcolor=FFFFFF>outline anti-aliasing</fgcolor></**sha**></font>*" gets:

The **Alignment** parameter can be one of the followings:

- 0 - exTopLeft
- 1 - exTopRight
- 2 - exBottomLeft
- 3 - exBottomRight
- 0x10 - exCenter
- 0x11 - exCenterLeft
- 0x12 - exCenterRight
- 0x13 - exCenterTop
- 0x14 - exCenterBottom

*Use numeric values as strings for X and Y parameters, to move the tooltip window relative to the position of the cursor. For instance, ShowToolTp("text",,,**"11"**,**"12"**), means that the tooltip window is moved 11 pixels on the X axis, and 12 pixels on the Y axis, before showing it in the default position. In this case the X and Y parameters MUST be passed as strings not as LONG values.*

The following VB sample displays the tooltip when the cursor hovers the form:

```
Private Sub Form_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
   t.ShowToolTip "This is a bit of text that's shown when the cursor hovers the form"
End Sub
```

The following VB.NET sample displays the tooltip when the cursor hovers the form:

```
Private Sub Form1_MouseMove(ByVal sender As Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles Me.MouseMove
   t.ShowToolTip("This is a bit of text that's shown when the cursor hovers the form")
End Sub
```

The following C# sample displays the tooltip when the cursor hovers the form:

```
private void Form1_MouseMove(object sender, MouseEventArgs e)
{
   t.ShowToolTip("This is a bit of text that's shown when the cursor hovers the form", "", "",
"", "" );
}
```

The following C++ sample displays the tooltip when the cursor hovers the form:

```
BOOL PreTranslateMessage(MSG* pMsg)
{
    /*
        Handles the WM_MOUSEMOVE message during the PreTranslateMessage so it can
show the tooltip even if we move the mouse over the inside controls too.
        As the WM_MOUSEMOVE message is not sent to dialog, if the cursor hovers the
inside windows...
    */
    if ( pMsg->message == WM_MOUSEMOVE )
    {
        if ( m_spToolTip != NULL )
            m_spToolTip->ShowToolTip( COleVariant( "This is a bit of text that's shown when
the cursor hovers the form" ), vtMissing, vtMissing, vtMissing );
    }
    return CDialog::PreTranslateMessage(pMsg);
}
```

The following VFP sample displays the tooltip when the cursor hovers the form:

```
LPARAMETERS nButton, nShift, nXCoord, nYCoord

with t
    .ShowToolTip("This is a bit of text that's shown when the cursor hovers the form")
endwith
```

property ToolTip.Template as String

Specifies the control's template.

Type	Description
String	A string expression that defines the control's template

The control's template uses the X-Script language to initialize the control's content. Use the Template property page of the control to update the control's Template property. Use the Template property to execute code by passing instructions as a string (template string). Use the ExecuteTemplate property to get the result of executing a template script.

The Exontrol's eXHelper tool helps you to find easy and quickly the answers and the source code for your questions regarding the usage of our UI components.

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 ToolTip.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 ToolTip.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 ToolTip.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 tooltip is never shown. 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 ToolTipHeight property to specify the height of the tooltip window. Use the ShowToolTip method to display the tooltip. Use the HideToolTip method to hide programmatically the tooltip. Use the Appearance property to specify the tooltip's appearance.

property ToolTip.ToolTipHeight as Long

Specifies a value that indicates the height of the ToolTip, in pixels.

Type	Description
Long	A long expression that specifies the height of the tooltip, when it displays a custom size picture.

By default, the ToolTipHeight property is 96 pixels. Use the ToolTipHeight property to change the tooltip window height. The height of the tooltip window is automatically computed based on tooltip's description. If the tooltip property displays a custom size picture, the ToolTipHeight property specifies the height of the tooltip. The ToolTipWidth property specifies a value that indicates the width of the ToolTip, in pixels. If the ToolTipDelay or ToolTipPopDelay property is 0, the tooltip is never shown. The ToolTipDelay property Specifies the time in ms that passes before the ToolTip appears. 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 ShowToolTip method to display the tooltip. Use the HideToolTip method to hide programmatically the tooltip. Use the Appearance property to specify the tooltip's appearance.

property ToolTip.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 tooltip is never shown. The ToolTipDelay property specifies the time in ms that passes before the ToolTip appears. 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 ToolTipHeight property to specify the height of the tooltip window. Use the ShowToolTip method to display the tooltip. Use the HideToolTip method to hide programmatically the tooltip. Use the Appearance property to specify the tooltip's appearance.

property ToolTip.ToolTipWidth as Long

Specifies a value that indicates the width of the ToolTip, in pixels.

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

By default, the ToolTipWidth property is 196 pixels. Use the ToolTipWidth property to change the tooltip window width. The height of the tooltip window is automatically computed based on tooltip's description. If the tooltip property displays a custom size picture, the ToolTipHeight property specifies the height of the tooltip. If the ToolTipDelay or ToolTipPopDelay property is 0, the tooltip is never shown. The ToolTipDelay property Specifies the time in ms that passes before the ToolTip appears. 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 ShowToolTip method to display the tooltip. Use the HideToolTip method to hide programmatically the tooltip. Use the Appearance property to specify the tooltip's appearance.

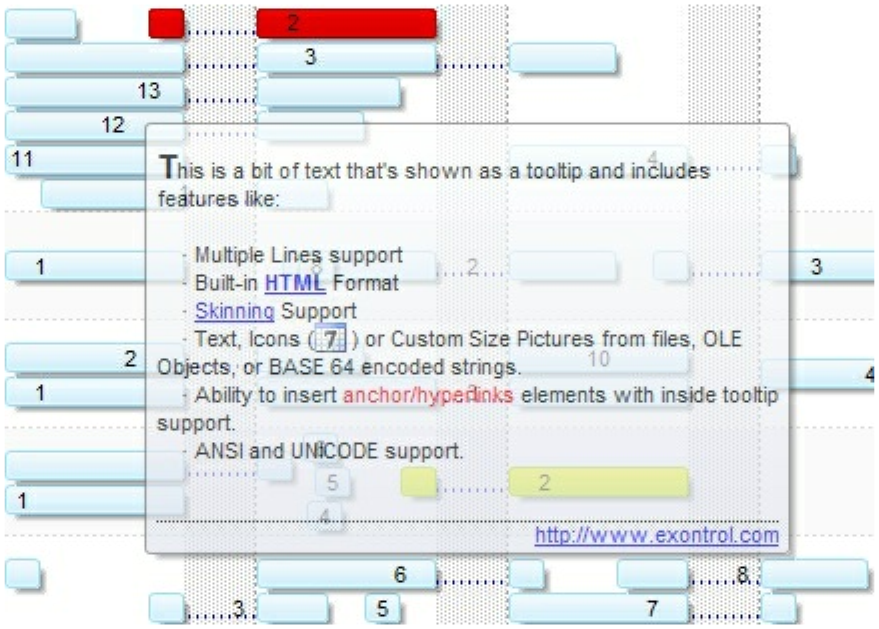
property ToolTip.Transparency as Long

Specifies the tooltip's transparency.

Type	Description
Long	A long expression that specifies the transparency of the tooltip, 0 means fully opaque, 50 means semi-transparent while 100 means completely transparent.

By default, the Transparency property is 0. The Transparency property has no effect if it is 0. Use the Transparency property to display semi-transparent tooltips. The TransparencyInside property specifies the transparency for inside tool tip if it is displayed. The Transparency property gets or sets a value indicating the percentage of transparency of the tooltip. The transparency is a value from 0 to 100; 0 indicating fully opaque and 100 indicating a completely transparent tooltip.

The following screen shot shows a semi-transparent tooltip (Transparency = 50):

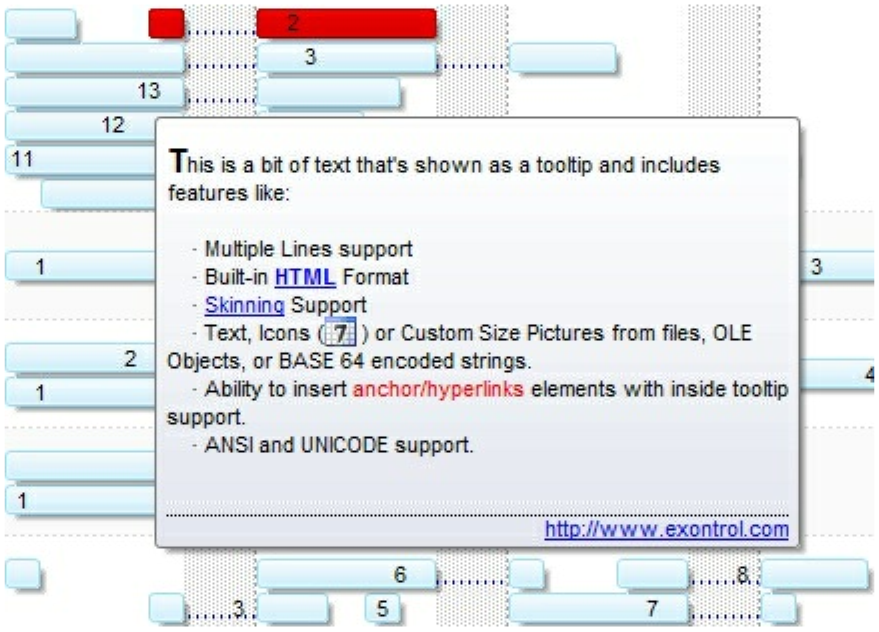


This is a bit of text that's shown as a tooltip and includes features like:

- Multiple Lines support
- Built-in HTML Format
- Skinning Support
- Text, Icons (7) or Custom Size Pictures from files, OLE Objects, or BASE 64 encoded strings.
- Ability to insert anchor/hyperlinks elements with inside tooltip support.
- ANSI and UNICODE support.

http://www.exontrol.com

The following screen shot shows an opaque tooltip (Transparency = 0, by default):


This is a bit of text that's shown as a tooltip and includes features like:

· Multiple Lines support
· Built-in **HTML** Format
· Skinning Support
· Text, Icons (7) or Custom Size Pictures from files, OLE Objects, or BASE 64 encoded strings.
· Ability to insert anchor/hyperlinks elements with inside tooltip support.
· ANSI and UNICODE support.

http://www.exontrol.com

property ToolTip.TransparencyInside as Long

Specifies the inside tooltip's transparency.

Type	Description
Long	A long expression that specifies the transparency of the inside tooltip, 0 means fully opaque, 50 means semi-transparent while 100 means completely transparent.

By default, the TransparencyInside property is 0. The TransparencyInside property has no effect if it is 0. Use the TransparencyInside property to display semi-transparent inside tooltips. Use the anchor elements to display inside tooltip. The TransparencyInside property gets or sets a value indicating the percentage of transparency of the inside tooltip. The transparency is a value from 0 to 100; 0 indicating fully opaque and 100 indicating a completely transparent tooltip.

property ToolTip.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 ToolTip.Visible as Boolean

Specifies whether the tooltip is visible.

Type	Description
Boolean	A Boolean expression that indicates whether the tooltip is visible or hiden

The Visible property determines if the tooltip is visible or hiden. Use the ShowToolTip method to show programmatically the tooltip. Use the HideToolTip method to hide programmatically the tooltip. Use the hWnd property to access the handle of the window that hosts the tooltip. If the ToolTipDelay or ToolTipPopDelay property is 0, the tooltip is never shown.

property ToolTip.VisualAppearance as Appearance

Retrieves the control's appearance.

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

Use the VisuapAppearance property to access the tooltip's Appearance collection. Use the Appearance property to specify the tooltip's border. Use the BackColor property to specify the tooltip's background color. Use the ForeColor property to specify the tooltip's foreground color.

The following VB sample changes the border of the tooltip, using an EBN file:

```
Private Sub Form_Load()
    Set t = New EXTOOLTIPLib.ToolTip
    With t
        .VisualAppearance.Add &H12, "c:\temp\winword.ebn"
        t.Appearance = &H12000000
        t.BackColor = RGB(255, 255, 255)
    End With
End Sub
```

The following VB.NET sample changes the border of the tooltip, using an EBN file:

```
Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles MyBase.Load
    t = New EXTOOLTIPLib.ToolTip
    t.VisualAppearance.Add(&H12, "c:\temp\winword.ebn")
    t.Appearance = &H12000000
    t.BackColor = ToUInt32(Color.White)
End Sub
```

where the ToUInt32 function is defined like follows:

```
Shared Function ToUInt32(ByVal c As Color) As UInt32
    Dim i As Long
    i = c.R
    i = i + 256 * c.G
    i = i + 256 * 256 * c.B
    ToUInt32 = Convert.ToUInt32(i)
End Function
```

The following C# sample changes the border of the tooltip, using an EBN file:

```
private void Form1_Load(object sender, EventArgs e)
{
    t = new EXTOOLTIPLib.ToolTip();
    t.VisualAppearance.Add(0x12, "c:\\temp\\winword.ebn");
    t.Appearance = (EXTOOLTIPLib.AppearanceEnum)0x12000000;
    t.BackColor = ToUInt32(Color.White);
}
```

where the ToUInt32 function is defined like follows:

```
private UInt32 ToUInt32(Color c)
{
    long i;
    i = c.R;
    i = i + 256 * c.G;
    i = i + 256 * 256 * c.B;
    return Convert.ToUInt32(i);
}
```

The following C++ sample changes the border of the tooltip, using an EBN file:

```
void initToolTip()
{
    CoInitialize( NULL );
    if ( SUCCEEDED( CoCreateInstance( __uuidof(EXTOOLTIPLib::ToolTip), NULL, CLSCTX_ALL,
__uuidof(EXTOOLTIPLib::IToolTip), (LPVOID*)&m_spToolTip ) ) )
    {
        m_spToolTip->VisualAppearance->Add( 0x12, COleVariant( "c:\\temp\\winword.ebn"
) );
        m_spToolTip->Appearance = (EXTOOLTIPLib::AppearanceEnum)0x12000000;
        m_spToolTip->BackColor = RGB(255,255,255);
    }
}
```

The following VFP sample changes the border of the tooltip, using an EBN file:

```
public t as Object
t = CreateObject("Exontrol.ToolTip")

with t
    .VisualAppearance.Add(0x12,"c:\temp\winword.ebn")
    .Appearance = 0x12000000
    .BackColor = RGB(255,255,255)
endwith
```

ExToolTip events

Adding the component to your projects is very easy, and requires only a few lines of code. The ToolTip object supports the following event(s):

Name	Description
AnchorClick	Occurs when an anchor element is clicked.

event AnchorClick (AnchorID as String, Options as String)

Occurs when an anchor element is clicked.

Type	Description
AnchorID as String	A string expression that indicates the identifier of the anchor
Options as String	A string expression that specifies options of the anchor element. *For the eXToolTip control, the Options parameter specifies the inside tooltip being shown when the cursor hovers 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</a>*, 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</a>*, the AnchorID parameter of the AnchorClick event is 1, and the Options parameter is "yourextradata".

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,
AxEXTOOLTIPLib._IToolTipEvents_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:
AxEXTOOLTIPLib._IToolTipEvents_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
AxEXTOOLTIPLib._IToolTipEvents_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(oToolTip,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 opens your default browser (IE, Chrome, Firefox, ...) when user clicks an hyperlink (using the /COM assembly version):

```
Private Sub Form_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
    If Not (t.Visible) Then
        t.ShowToolTip "<a www.exontrol.com>exontrol</a> click and so on", Nothing,
EXTOOLTIPLib.exTopLeft, "+16", Nothing
    End If
End Sub

Private Sub t_AnchorClick(ByVal AnchorID As String, ByVal Options As String)
```

```
    Shell ("C:\Program Files\Internet Explorer\IEXPLORE.EXE " & "http://" & AnchorID)
End Sub
```

The following VB/NET sample opens your default browser (IE, Chrome, Firefox, ...) when user clicks an hyperlink (using the /NET assembly version):

```
Private Sub Form1_MouseMove(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles MyBase.MouseMove
    If Not (Extooltip1.Visible) Then
        Extooltip1.ShowToolTip("<a www.exontrol.com>exontrol</a> click and so on",
Nothing, exontrol.EXTOOLTIPLib.AlignmentEnum.exTopLeft, "+16", Nothing)
    End If
End Sub

Private Sub Extooltip1_AnchorClick_1(ByVal sender As System.Object, ByVal AnchorID As
System.String, ByVal Options As System.String) Handles Extooltip1.AnchorClick
    System.Diagnostics.Process.Start("http://" + AnchorID.ToString())
End Sub
```

The AnchorID parameter carries the first argument of the <a param1;param2> HTML tag in the tooltip. In the previously sample, the first parameter of the <a> is www.exontrol.com which is passed to AnchorClick event when user presses the exontrol link in the tooltip.