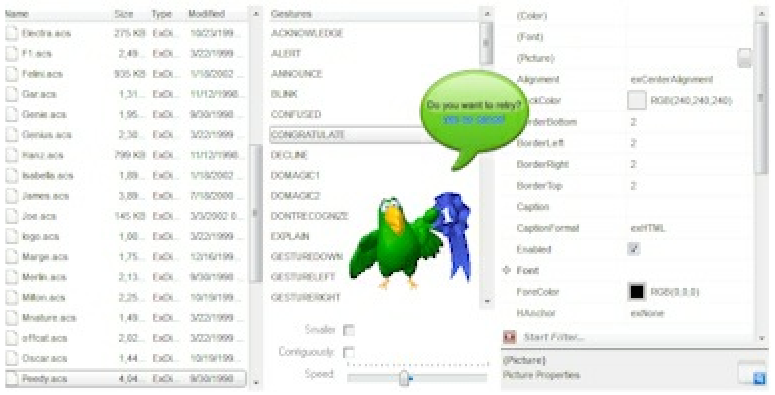
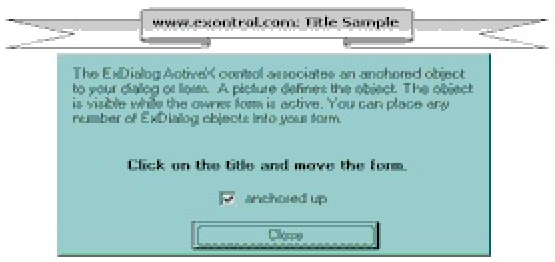
ExDialog

Exontrol's Agent (eXDialog) component lets you add personalities to your application, similar with the Microsoft Agent or Microsoft Office Assistant. All you need is the file that stores the animated character, (such as ACS/ACF files), and a call like Play("peedy.acs","Congratulate","repeat") which shows the "Congratulate" animation from "peedy.acs" file. ACS/ACF is a file extension for an animated character file used by Microsoft Agent. The Exontrol's Agent component requires no third party libraries, so no Microsoft Agent Server, Microsoft Agent Control components are required.

Features of eXAgent/eXDialog include:

- **ACS/ACF** Files Support. Microsoft Agent characters are stored in files of the .ACS extension, and can be stored in a number of compressed .ACF files for better World Wide Web distribution
- **HTML** Caption Support
- **no** third library required




www.exontrol.com: Title Sample

The ExDialog ActiveX control associates an anchored object to your dialog or form. A picture defines the object. The object is visible while the owner form is active. You can place any number of ExDialog objects into your form.

Click on the title and move the form.

☑ anchored up

Close

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

How to get support?

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

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

- Check out the samples - they are here to provide some quick info on how things should be done
- Check out the how-to questions using the eXHelper tool
- Check out the help - includes documentation for each method, property or event
- Check out if you have the latest version, and if you don't have it send an update request here.
- Submit your problem(question) here.

Don't forget that you can contact our development team if you have ideas or requests for new components, by sending us an e-mail at support@exontrol.com (please include the name of the product in the subject, ex: exgrid) . We're sure our team of developers will try to find a way to make you happy - and us too, since we helped.

Regards,
Exontrol Development Team

https://www.exontrol.com

constants AlignmentEnum

The AlignmentEnum type specifies the way the control's caption is horizontally aligned. The Alignment property specifies the horizontal alignment for the control's caption. The AlignmentEnum type supports the following values:

Name	Value	Description
exLeftAlignment	0	Aligns the caption to the left.
exCenterAlignment	1	Centers the caption.
exRightAlignment	2	Aligns the caption to the right.

constants AnchorEnum

The ExDialog control can be anchored to the container edges.

Name	Value	Description
exNone	0	None. The control is not anchored.
exLeft	1	exLeft. The control is anchored to the left edge of its container.
exRight	2	exRight. The control is anchored to the right edge of its container.
exCenter	3	exCenter. The control is anchored to the midle edge of its container.
exTop	17	exTop. The control is anchored to the top edge of its container.
exBottom	18	exBottom. The control is anchored to the bottom edge of its container.
exMiddle	19	exMiddle. The control is anchored to the midle edge of its container.

constants CaptionFormatEnum

Specifies the format of the control's caption.

Name	Value	Description
exText	0	The Caption property displays no HTML format.
		The control uses built-in HTML tags to display the caption using HTML format. The control supports the following HTML tags: • **<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.

exHTML	1	• **<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:

outline anti-aliasing

constants ClickActionEnum

Specifies what the control does when the user clicks.

Name	Value	Description
exNothing	0	Nothing.
exMoveContainer	1	Moves the container, if the user clicks and drags the control.
exMoveControl	2	Moves the control, if the user clicks and drags the control.

constants VAlignmentEnum

The VAlignmentEnum type specifies the way the control's caption is vertically aligned. The VAlignment property specifies the control's caption vertical alignment. The VAlignmentEnum type supports the following values:

Name	Value	Description
exTopAlignment	0	Aligns the caption to the top.
exMiddleAlignment	4	Centers the caption.
exBottomAlignment	8	Aligns the caption to the bottom.

Dialog 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: {2775B67A-6CFC-4ADC-9E73-A7CA6CAA2844}. The object's program identifier is: "Exontrol.Dialog". The /COM object module is: "ExDialog.dll"

The Dialog object associates a picture to a form. The object supports the following properties and methods:

Name	Description
Alignment	Aligns the control's caption.
Attach	Attaches the control to the container.
BackColor	Retrieves or sets the control's background color.
BorderBottom	Retrieves or sets the size in pixels of the border on the bottom side.
BorderLeft	Retrieves or sets the size in pixels of the border on the left side.
BorderRight	Retrieves or sets the size in pixels of the border on the right side.
BorderTop	Retrieves or sets the size in pixels of the border on the up side.
Caption	Retrieves or sets the control's caption.
CaptionFormat	Specifies whether the control's caption displays data using HTML format.
Continue	Continues the gesture/animation.
Dettach	Detaches the control.
Enabled	Retrieves or sets a value indicating whether the control is visible or hidden.
Font	Retrieves or sets the control's font.
ForeColor	Retrieves or sets the control's foreground color.
HAnchor	Retrieves or sets a value that indicates how the control is horizontally anchored.
HAnchorOffset	Retrieves or sets a value that indicates the horizontally offset of the agent control.
HCaptionOffset	Retrieves or sets a value that indicates the horizontally offset of the agent's caption.
HTMLPicture	Adds or replaces a picture in HTML captions.

hWnd	Retrieves the control's window handle.
OnClick	Retrieves or sets a value that indicates what the action that control takes when the user clicks on the control.
Picture	Retrieves or sets the control's picture.
PictureHeight	Retrieves the picture's height.
PictureWidth	Retrieves the picture's width.
Play	Plays a gesture/animation from giving agent file.
Refresh	Refreshes the control.
Replay	Replays the gesture/animation.
Stop	Stops the gesture/animation.
TopMost	Places the control above all non-topmost windows.
Transparency	Retrieves or sets a value that indicates whether the control allows transparency.
TransparentFrom	Retrieves or sets a value that indicates the starting transparent color.
TransparentTo	Retrieves or sets a value that indicates the ending transparent color.
VAlignment	Aligns vertically the control's caption.
VAnchor	Retrieves or sets a value that indicates how the control is vertically anchored.
VAnchorOffset	Retrieves or sets a value that indicates the vertically offset of the agent control.
VCaptionOffset	Retrieves or sets a value that indicates the vertically offset of the agent's caption.
Version	Retrieves the control's version.

property Dialog.Alignment as AlignmentEnum

Aligns horizontally the control's caption.

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

By default, the Alignment property is exLeft. The Caption property defines the control's caption. Use the CaptionFormat property to specify whether the control displays its caption using HTML format. The VAlignment property defines the vertical alignment for the control's caption. The HCaptionOffset property defines the horizontal offset of the control's caption. The VCaptionOffset property defines the vertical offset of the control's caption.

method Dialog.Attach ([Reserved as Variant])

Attaches the control to the container.

Type	Description
Reserved as Variant	A Long expression that defines the handle of the window to attach the agent control. By attaching it means that the agent is visible, while its container is visible or active window.

Use the Attach method to attach the control to the container. Usually, you can call call Attach method when the form is loading (Load event). Use the Dettach method to detach the control. Use the Enabled property to show or hide the control. Use the Picture property to associate a picture to the agent control. Use the Play method play an ACS/ACF file. Use the Caption property to define the control's caption. Warning! If the form contains multiple instance of ExDialog controls, you have to call Dettach method in reverse order.

For instance, if you called:

```
Private Sub Form_Load()
   Dialog1.Attach
   Dialog2.Attach
End Sub
```

call like follows to detach the controls:

```
Private Sub Form_Unload(Cancel As Integer)
   Dialog2.Dettach
   Dialog1.Dettach
End Sub
```

property Dialog.BackColor as Color

Retrieves or sets the control's background color.

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

The background color has effect only if the loaded picture is an icon. The BackColor property has no effect if the Picture property points to an icon file. Use the TransparentFrom and TransparentTo properties to define the transparent colors of the control.

property Dialog.BorderBottom as Long

Retrieves or sets the size in pixels of the border on the bottom side.

Type	Description
Long	A long expression that defines the border size on the left side.

The property has no effect if the control's Caption property is empty. Use the BorderLeft, BorderTop, BorderRight and BorderBottom properties to define the control's caption borders.

property Dialog.BorderLeft as Long

Retrieves or sets the size in pixels of the border on the left side.

Type	Description
Long	A long expression that indicates the control's caption border size on the left side.

The property has no effect if the control's Caption property is empty. Use the BorderLeft, BorderTop, BorderRight and BorderBottom properties to define the control's caption borders.

property Dialog.BorderRight as Long

Retrieves or sets the size in pixels of the border on the right side.

Type	Description
Long	A long expression that defines the right border size of the control's caption.

The property has no effect if the control's Caption property is empty. Use the BorderLeft, BorderTop, BorderRight and BorderBottom properties to define the control's caption borders.

property Dialog.BorderTop as Long

Retrieves or sets the size in pixels of the border on the up side.

Type	Description
Long	A long expression that defines the top border size of the control's caption.

The property has no effect if the control's Caption property is empty. Use the BorderLeft, BorderTop, BorderRight and BorderBottom properties to define the control's caption borders.

property Dialog.Caption as String

Retrieves or sets the control's caption.

Type	Description
String	A string expression that defines the control's caption. The Caption may include HTML tags. Use the CaptionFormat property to specify whether the control displays its caption using HTML format.

By default, the Caption property is empty. Use the Caption property to define the control's caption. Use the Picture property to assign a picture to the control. **The Caption property has no effect if the control has no picture associated.**

Use the Paint event to allow owner draw control. The Caption of the control uses the BorderLeft, BorderTop, BorderRight and BorderBottom properties to define the rectangle where it is painted. The Font property defines the font used to paint the caption. The ForeColor property specifies the caption's foreground color. Use the Alignment property to align the control's caption. Use the CaptionFormat property to specify the way how text is displayed. The VAlignment property defines the vertical alignment for the control's caption. The HCaptionOffset property defines the horizontal offset of the control's caption. The VCaptionOffset property defines the vertical offset of the control's caption.

The following sample displays a HTML text:

```
With Dialog1
    .Caption = "This <b>is</b> <fgcolor=00FF00>just</fgcolor> a <r>text that should
break the line. Isn't it so?"
    .CaptionFormat = exHTML
End With
```

property Dialog.CaptionFormat as CaptionFormatEnum

Specifies whether the control's caption displays data using HTML format.

Type	Description
CaptionFormatEnum	A CaptionFormatEnum expression that indicates the format of the control's caption.

By default, the CaptionFormat is exText. The control displays the control's Caption using HTML format only if the CaptionFormat property is exHTML. The Alignment property defines the horizontal alignment for the control's caption. The VAlignment property defines the vertical alignment for the control's caption. The HCaptionOffset property defines the horizontal offset of the control's caption. The VCaptionOffset property defines the vertical offset of the control's caption.

The caption supports the following HTML tags:

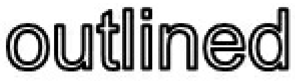
- **<b> ... </b>** displays the text in **bold**
- **<i> ... </i>** displays the text in *italics*
- **<u> ... </u>** underlines the text
- **<s> ... </s>** ~~Strike~~-through text
- **<a id;options> ... </a>** displays an anchor element that can be clicked. An anchor is a piece of text or some other 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.

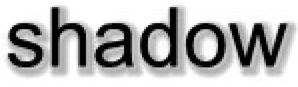
- **<upline> ... </upline>** draws the line on the top side of the current text-line (requires <solidline> or <dotline>).
- **<r>** right aligns the text
- **<c>** centers the text
- **<br>** forces a line-break
- **<img>number[:width]</img>** inserts an icon inside the text. The number indicates the 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

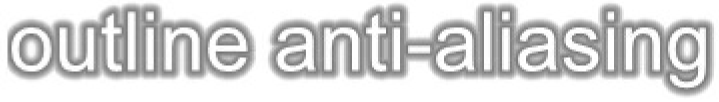
- **<out rrggbb;width> ... </out>** shows the text with outlined characters, where rr/gg/bb represents the red/green/blue values of the outline color, 808080 if missing as gray, width indicates the size of the outline, 1 if missing. The text color or <fgcolor> defines the color to show the inside text.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



method Dialog.Continue ()

Continues the gesture/animation.

Type	Description

Use the Continue method to continue playing the current gesture. The Continue method has no effect if no ACS/ACF file was previously loaded using the Play method. The Stop method stops animation of the current gesture. Use the Replay method to start re-playing the gesture(s) from the beginning. The Picture property is changed periodically when a new frame from the current gesture is shown.

method Dialog.Dettach ()

Detaches the control.

Type	Description

Call Dettach method to detach the control. If the control is attached to the container, use Dettach method to detach the control. Use the Attach method to attach a control to a form. If the form contains multiple instances of the ExDialog object, you have to call Dettach method in reverse order.

property Dialog.Enabled as Boolean

Retrieves or sets a value indicating whether the control is visible or hidden.

Type	Description
Boolean	A boolean expression indicating whether the control is visible or hidden.

Use the Enabled property to show or hide the control. The Visible property (that's implemented by the container) has no effect for ExDialog sample.

property Dialog.Font as IFontDisp

Retrieves or sets the control's font.

Type	Description
IFontDisp	A Font object that defines the caption's font.

Use the Font property to define the control's caption font. Use the Caption property to specify the control's caption.

property Dialog.ForeColor as Color

Retrieves or sets the control's foreground color.

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

The ForeColor property has effect only if the Caption property is not empty.

property Dialog.HAnchor as AnchorEnum

Retrieves or sets a value that indicates how the control is horizontally anchored.

Type	Description
AnchorEnum	An AnchorEnum expression that indicates how the control is horizontally anchored.

Use the HAnchor and VAnchor property to control to container edges. If the control is anchored, the HAnchorOffect and VAnchorOffset properties defines the control's offset to anchored edge. If the control is not anchored, the HAnchorOffect and VAnchorOffset properties defines the control's position in screen coordinates. Use the OnClick property to define the action that control executes when the user clicks on the control.

property Dialog.HAnchorOffset as Long

Retrieves or sets a value that indicates the horizontally offset.

Type	Description
Long	A long expression that defines the control's offset to container edges.

If the HAnchor property is exNone, the HAnchorOffset property specifies the control's horizontal position, in screen coordinates. If the HAnchor property is not exNone, the HAnchorOffset property defines the control's offset to anchored edge. Use the OnClick event property to define the action that control executes when the user clicks on the control. The Move event is called if the container is moved or resized.

property Dialog.HCaptionOffset as Long

Retrieves or sets a value that indicates the horizontally offset of the agent's caption.

Type	Description
Long	A long expression that specifies the horizontal offset of the control's caption.

By default, the HCaptionOffset property is 0. The HCaptionOffset property defines the horizontal offset of the control's caption. Use the Caption property to specify the control's caption. The VCaptionOffset property defines the vertical offset of the control's caption. The Caption of the control uses the BorderLeft, BorderTop, BorderRight and BorderBottom properties to define the rectangle where it is painted. The Font property defines the font used to paint the caption. The ForeColor property specifies the caption's foreground color. Use the Alignment property to align the control's caption. Use the CaptionFormat property to specify the way how text is displayed. The VAlignment property defines the vertical alignment for the control's caption.

property Dialog.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 sample shows how to put a custom size picture in the column's header:

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


<CONTROL>.Caption = "A <img>pic1</img>"
<CONTROL>.Caption = "B <img>pic2</img>"
<CONTROL>.Caption = "A <img>pic1</img> + B <img>pic2</img>"
```

property Dialog.hWnd as Long

Retrieves the control's window handle.

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

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

property Dialog.OnClick as ClickActionEnum

Retrieves or sets a value that indicates what the action that control takes when the user clicks on the control.

Type	Description
ClickActionEnum	A ClickActionEnum expression that defines the action that control executes when the user clicks the control.

Use the OnClick property to define what control should do if the user clicks on the control. If the OnClick property is exMoveContainer or exMoveControl, the Click or RClick event is not fired if the user clicked the control and it starts to move the window. The Click or RClick is fired in these cases only if the user doesn't move the window.

property Dialog.Picture as IPictureDisp

Retrieves or sets the control's picture.

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

Use the Picture property to define the control's picture. The Play method plays animation from specified ACS/ACF file. Use the Enabled property to show or hide the agent/assistant control. Use the Transparency property to specify whether the picture is shown using transparent colors. Use the TransparentFrom and TransparentTo properties to define the transparent colors. The control's size is defined by the control's picture. If the control has no picture the control's size is empty. Use the BackColor property to define the control's background color if the Picture points to an icon file. Use the Caption property to specify the control's caption. The PictureChanged event occurs once the Picture property is changed. **The Picture property has no effect if the Attach method is not called.**

property Dialog.PictureHeight as Long

Retrieves the picture's height.

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

If the control has no picture loaded, the PictureHeight property gets 0. Use the PictureWidth property to retrieve the picture's width.

property Dialog.PictureWidth as Long

Retrieves the picture's width.

Type	Description
Long	A long expression that indicates the picture's width in pixels.

Use the PictureWidth property to get the picture's width. Use the PictureHeight property to get the picture's height. If the control has no picture loaded, the PictureWidth property retreves 0.

method Dialog.Play (AgentFile as Variant, [Gesture as Variant], [Options as Variant])

Plays a gesture/animation from giving agent file.

Type	Description
AgentFile as Variant	A String expression that specifies the ACS or ACF file to be loaded. Microsoft Agent characters are stored in files of the .ACS extension, and can be stored in a number of compressed .ACF files for better World Wide Web distribution.
Gesture as Variant	A String expression that defines the name of the gesture to play, or a list of gestures to be played. The Gesture parameter supports patterns with wild characters. The space character splits multiple patterns. The Gesture may include a pattern with wild characters as *,?,# or []. For instance, "*" specifies all gestures, while "S*" specifies all gestures that start with character S. **Each ACS/ACF file provides its list of gestures.**
Options as Variant	A String expression that defines the options to play the gesture(s). The options are separated by comma characters. The option supports any of the following: • **repeat**, the gesture(s) plays contiguously. • **speed=value**, defines the speed to play the gesture/animation. The value is a positive float number. The 0 is the faster, while a bigger value specifies to play slower the animation. By default, the value is 10. • **size=value**, defines the size of the gesture. The value is a positive float number. By default, value is 1, which makes the gesture to show its normal size. • **stop**, indicates that the Play method does not start playing. *For instance:* • *Play "Peedy.acs", "Show" loads the peedy.acs animation file, and starts playing the Show gesture.* • *Play "Peedy.acs", "LookDown LookDownReturn" loads the peedy.acs animation file, and starts playing the LookDown gesture, and after the LookDown is*

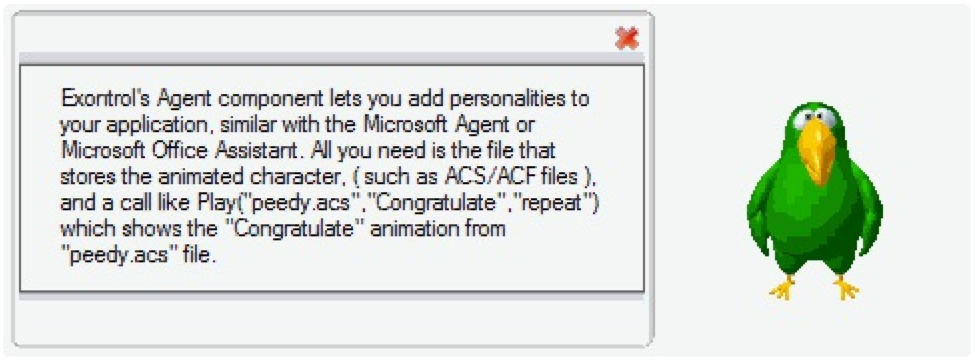
finish, the LookDownReturn animation is played.

- *Play "Peedy.acs", "L*", "repeat" loads the peedy.acs animation file, and starts playing repeatedly all gestures that starts with L.*
- *Play "Peedy.acs", "*", "repeat,size=2,speed=0" loads the peedy.acs animation file, and starts playing repeatedly all gestures, as fast as possible, and showing double the pictures.*
- *Play "Peedy.acs","*","stop" gets the list of all gestures in the peedy.acs file. The stop indicates that the animation is not started.*

Return	Description
Variant	A String expression that defines the list of gestures to be played separated by comma character. *For instance, the Play("peedy.acs","*","stop") gets the list of all gestures in the peedy.acs file.*

The Play method plays a gesture or a list of gestures from giving file. The Play method returns the list of gestures to be played. The Play method starts playing the gestures, if any, while the "stop" option is not included in the Options parameter. Use the Stop method to stop playing the current gesture. Use the Continue method to continue playing the current gesture. Use the Replay method to start re-playing the gesture(s) from the beginning. The Picture property is changed periodically when a new frame from the current gesture is shown. The PictureChanged event notifies once the Picture property is changed. Use the Enabled property to show or hide the agent/assistant control. Use the Transparency property to specify whether the picture is shown using transparent colors. By default, the transparent color of the animation is defined by the ACS/ACF file, but you can use the TransparentFrom and TransparentTo properties to define the new transparent colors. **The Play method has no effect if the Attach method is not called.**

By default, the transparent color of the animation is defined by the ACS/ACF file, but you can use the TransparentFrom and TransparentTo properties to define the new transparent colors.


Exontrol's Agent component lets you add personalities to your application, similar with the Microsoft Agent or Microsoft Office Assistant. All you need is the file that stores the animated character, (such as ACS/ACF files), and a call like Play("peedy.acs","Congratulate","repeat") which shows the "Congratulate" animation from "peedy.acs" file.

method Dialog.Refresh ()

Refreshes the control.

Type	Description

The Refresh method refreshes the control.

method Dialog.Replay ()

Replays the gesture/animation.

Type	Description

Use the Replay method to start re-playing the gesture(s) from the beginning. The Replay method has no effect if no ACS/ACF file was previously loaded using the Play method. Use the Continue method to continue playing the current gesture. The Stop method stops animation of the current gesture. The Picture property is changed periodically when a new frame from the current gesture is shown.

method Dialog.Stop ()

Stops the gesture/animation.

Type	Description

The Stop method stops animation of the current gesture. The Stop method has no effect if no ACS/ACF file was previously loaded using the Play method. Use the Continue method to continue playing the current gesture. Use the Replay method to start re-playing the gesture(s) from the beginning. The Picture property is changed periodically when a new frame from the current gesture is shown.

property Dialog.TopMost as Boolean

Places the control above all non-topmost windows.

Type	Description
Boolean	A boolean expression that defines whether the control is top-most or not.

Use the TopMost property to put the control on top.

property Dialog.Transparency as Boolean

Retrieves or sets a value that indicates whether the control allows transparency.

Type	Description
Boolean	A boolean expression that indicates whether the control allows transparency.

If the Transparency property is true, the TransparentFrom and TransparentTo properties define the transparent color. Use the Transparecy property to disable transparent colors for the control.

property Dialog.TransparentFrom as Color

Retrieves or sets a value that indicates the starting transparent color.

Type	Description
Color	A color expression that indicates the starting transparent color.

Use the TransparentFrom and TransparentTo properties to define the transparent colors for the control. These properties have efect only if the Transparency property is True. Use the Picture property to load a picture into the control. If the Picture property points to an icon, the BackColor property defines the control's background color. Use the same value for TransparentFrom and TransparentTo as BackColor property, if you need to define a non-regular window based on an icon. By default, the transparent color of the animation (Play method) is defined by the ACS/ACF file, but you can use the TransparentFrom and TransparentTo properties to define the new transparent colors.

property Dialog.TransparentTo as Color

Retrieves or sets a value that indicates the transparent color to.

Type	Description
Color	A color expression that defines the ending transparent color.

Use the TransparentFrom and TransparentTo properties to define the transparent colors for the control. These properties have efect only if the Transparency property is True. Use the Picture property to load a picture into the control. If the Picture property points to an icon, the BackColor property defines the control's background color. Use the same value for TransparentFrom and TransparentTo as BackColor property, if you need to define a non-regular window based on an icon. By default, the transparent color of the animation (Play method) is defined by the ACS/ACF file, but you can use the TransparentFrom and TransparentTo properties to define the new transparent colors.

property Dialog.VAlignment as VAlignmentEnum

Aligns vertically the control's caption.

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

By default, the VAlignment property is exTop. The Caption property defines the control's caption. Use the CaptionFormat property to specify whether the control displays its caption using HTML format. The Alignment property defines the horizontal alignment for the control's caption. The HCaptionOffset property defines the horizontal offset of the control's caption. The VCaptionOffset property defines the vertical offset of the control's caption.

property Dialog.VAnchor as AnchorEnum

Retrieves or sets a value that indicates how the control is vertically anchored.

Type	Description
AnchorEnum	An AnchorEnum expression that indicates how the control is vertically anchored.

Use the HAnchor and VAnchor property to control to container edges. If the control is anchored, the HAnchorOffect and VAnchorOffset properties defines the control's offset to anchored edge. If the control is not anchored, the HAnchorOffect and VAnchorOffset properties defines the control's position in screen coordinates. Use the OnClick property to define the action that control executes when the user clicks on the control.

property Dialog.VAnchorOffset as Long

Retrieves or sets a value that indicates the vertically offset.

Type	Description
Long	A long expression that indicates the control's offset to container edge, if the control is anchored, or control's position if it is not anchored.

If the VAnchor property is exNone, the VAnchorOffset property specifies the control's vertical position, in screen coordinates. If the VAnchor property is not exNone, the VAnchorOffset property defines the control's offset to anchored edge. Use the OnClick event property to define the action that control executes when the user clicks on the control. The Move event is called if the container is moved or resized.

property Dialog.VCaptionOffset as Long

Retrieves or sets a value that indicates the vertically offset of the agent's caption.

Type	Description
Long	A long expression that specifies the vertical offset of the control's caption.

By default, the VCaptionOffset property is 0. The VCaptionOffset property defines the vertical offset of the control's caption. Use the Caption property to specify the control's caption. The HCaptionOffset property defines the horizontal offset of the control's caption. The Caption of the control uses the BorderLeft, BorderTop, BorderRight and BorderBottom properties to define the rectangle where it is painted. The Font property defines the font used to paint the caption. The ForeColor property specifies the caption's foreground color. Use the Alignment property to align the control's caption. Use the CaptionFormat property to specify the way how text is displayed. The VAlignment property defines the vertical alignment for the control's caption.

property Dialog.Version as String

Retrieves the control's version.

Type	Description
String	A String expression that indicates the version of the control you are running.

The Version property indicates the version of the control/dll you are running.

ExDialog 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: {2775B67A-6CFC-4ADC-9E73-A7CA6CAA2844}. The object's program identifier is: "Exontrol.Dialog". The /COM object module is: "ExDialog.dll"

The ExDialog ActiveX control supports the following events:

Name	Description
AnchorClick	Occurs when an anchor element is clicked.
Click	Occurs when the user clicks the control.
Leave	Notifies that the mouse is leaving the control's container.
MouseDown	Occurs when the user presses a mouse button.
MouseLeave	Notifies that the mouse is leaving the control.
MouseMove	Occurs when the user moves the mouse over the control.
MouseUp	Occurs when the user releases a mouse button.
Move	Occurs when the control's container is moved or resized.
Paint	Fired when the control is painting.
PictureChanged	Notifies your application once the control's Picture property is changed.
RClick	Occurs when the user right clicks the control.
Show	Occurs whether the control is shown or hidden.

event AnchorClick (AnchorID as String, Options as String)

Occurs when an anchor element is clicked.

Type	Description
AnchorID as String	A string expression that indicates the identifier of the anchor.
Options as String	A string expression that specifies options of the anchor element.

The control fires the AnchorClick event to notify that the user clicks an anchor element. An anchor is a piece of text or some other object (for example an image) which marks the beginning and/or the end of a hypertext link. The **<a>** element is used to mark that piece of text (or inline image), and to give its hypertextual relationship to other documents. The AnchorClick event is fired only if prior clicking the control it shows the hand cursor.

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,
AxEXDIALOGLib._IDialogEvents_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:
AxEXDIALOGLib._IDialogEvents_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
AxEXDIALOGLib._IDialogEvents_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(oDialog,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
```

event Click ()

Occurs when the user clicks the control.

Type	Description

Use the Click event to notify your application that user clicks on the control. Use RClick event to notify your application that the user has clicked the right mouse button. For instance, if the OnClick property is exMoveContainer or exMoveControl, the Click event is fired only if the user has not moved the control. The AnchorClick event occurs when the user clicks a hyperlink element (<a>).

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(oDialog)
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 Leave ()

Occurs whether the mouse leaves the container.

Type	Description

Use the Leave event to notify your application that the mouse leaves the container's client area. Use the MouseLeave event to notify your application that the mouse leaves the control's client area. Use the MouseMove event to notify your application that the mouse is moving over the control.

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

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

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

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

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

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

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

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

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

Powe…

```
begin event Leave()
end event Leave
```

VB.NET

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

VB6

```
Private Sub Leave()
End Sub
```

VBA

```
Private Sub Leave()
End Sub
```

VFP

```
LPARAMETERS nop
```

Xbas…

```
PROCEDURE OnLeave(oDialog)
RETURN
```

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

Java…

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

VBSc…

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

Visual Data…

```
Procedure OnComLeave
    Forward Send OnComLeave
End_Procedure
```

Visual Objects

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

X++

```
void onEvent_Leave()
```

```
{
}
```

XBasic

```
function Leave as v ()
end function
```

dBASE

```
function nativeObject_Leave()
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 screen 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 screen 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 event, MouseDown and MouseUp events lets you distinguish between the left, right, and middle mouse buttons. You can also write code for mouse-keyboard combinations that use the SHIFT, CTRL, and ALT keyboard modifiers

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

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

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

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

C#
```
private void MouseDownEvent(object sender,
AxEXDIALOGLib._IDialogEvents_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:
AxEXDIALOGLib._IDialogEvents_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
AxEXDIALOGLib._IDialogEvents_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(oDialog,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.Dialog.1::OLE_XPOS_PIXELS,Y as
OLE::Exontrol.Dialog.1::OLE_YPOS_PIXELS)
end function
```

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

event MouseLeave ()

Occurs when the mouse leaves the control.

Type	Description

Use the MouseLeave event to notify your application that the mouse leaves the control's client area. Use the Leave event to notify your application that the mouse leaves the container's client area.

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

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

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

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

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

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

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

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

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

Powe…
```
begin event MouseLeave()
end event MouseLeave
```

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

VB6
```
Private Sub MouseLeave()
End Sub
```

VBA
```
Private Sub MouseLeave()
End Sub
```

VFP
```
LPARAMETERS nop
```

Xbas…
```
PROCEDURE OnMouseLeave(oDialog)
RETURN
```

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

Java…
```
<SCRIPT EVENT="MouseLeave()" LANGUAGE="JScript">
</SCRIPT>
```

VBSc…
```
<SCRIPT LANGUAGE="VBScript">
Function MouseLeave()
End Function
</SCRIPT>
```

Visual Data…
```
Procedure OnComMouseLeave
   Forward Send OnComMouseLeave
End_Procedure
```

Visual Objects
```
METHOD OCX_MouseLeave() CLASS MainDialog
RETURN NIL
```

X++
```
void onEvent_MouseLeave()
{
}
```

XBasic

```
function MouseLeave as v ()
end function
```

dBASE

```
function nativeObject_MouseLeave()
return
```

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 over the control.

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 screen 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 OnClick property if you want to allow moving the control or the container while clicking the control.

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, AxEXDIALOGLib._IDialogEvents_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: AxEXDIALOGLib._IDialogEvents_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 AxEXDIALOGLib._IDialogEvents_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(oDialog,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.Dialog.1::OLE_XPOS_PIXELS,Y as
OLE::Exontrol.Dialog.1::OLE_YPOS_PIXELS)
end function
```

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

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 screen 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 screen 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 event, MouseDown and MouseUp events lets you distinguish between the left, right, and middle mouse buttons. You can also write code for mouse-keyboard combinations that use the SHIFT, CTRL, and ALT keyboard modifiers

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

C#

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

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,
AxEXDIALOGLib._IDialogEvents_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:
AxEXDIALOGLib._IDialogEvents_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
AxEXDIALOGLib._IDialogEvents_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(oDialog,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.Dialog.1::OLE_XPOS_PIXELS,Y as
OLE::Exontrol.Dialog.1::OLE_YPOS_PIXELS)
end function
```

dBASE

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

event Move (ByRef X as Long, ByRef Y as Long)

Occurs when the control's container is moved or resized.

Type	Description
X as Long	(By Reference) A single that specifies the current X location of the mouse pointer. The x values is always expressed in screen coordinates.
Y as Long	(By Reference) A single that specifies the current Y location of the mouse pointer. The y values is always expressed in screen coordinates

The Move event occurs when the control's position is changed. The Move event is called every time when the container is moved or resized. The X and Y coordinates define the control's position not the container position. The control is moved while the container is moved if the control is anchored to one of the container edges. To anchor the control to container edges use the HAnchor and VAnchor properties.

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

C#
```
private void Move(object sender,ref int X,ref int Y)
{
}
```

VB
```
Private Sub Move(ByVal sender As System.Object,ByRef X As Integer,ByRef Y As
Integer) Handles Move
End Sub
```

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

C#
```
private void Move(object sender, AxEXDIALOGLib._IDialogEvents_MoveEvent e)
{
}
```

C++
```
void OnMove(long FAR* X,long FAR* Y)
{
}
```

C++ Builder
```
void __fastcall Move(TObject *Sender,long * X,long * Y)
{
}
```

Delphi
```
procedure Move(ASender: TObject; var X : Integer;var Y : Integer);
begin
end;
```

Delphi 8 (.NET only)
```
procedure Move(sender: System.Object; e:
AxEXDIALOGLib._IDialogEvents_MoveEvent);
begin
end;
```

Powe…
```
begin event Move(long X,long Y)
end event Move
```

VB.NET
```
Private Sub Move(ByVal sender As System.Object, ByVal e As
AxEXDIALOGLib._IDialogEvents_MoveEvent) Handles Move
End Sub
```

VB6
```
Private Sub Move(X As Long,Y As Long)
End Sub
```

VBA
```
Private Sub Move(X As Long,Y As Long)
End Sub
```

VFP
```
LPARAMETERS X,Y
```

Xbas…
```
PROCEDURE OnMove(oDialog,X,Y)
RETURN
```

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

Java…
```
<SCRIPT EVENT="Move(X,Y)" LANGUAGE="JScript">
</SCRIPT>
```

VBSc…
```
<SCRIPT LANGUAGE="VBScript">
Function Move(X,Y)
End Function
</SCRIPT>
```

Visual Data...

```
Procedure OnComMove Integer llX Integer llY
    Forward Send OnComMove llX llY
End_Procedure
```

Visual Objects

```
METHOD OCX_Move(X,Y) CLASS MainDialog
RETURN NIL
```

X++

```
void onEvent_Move(COMVariant /*long*/ _X,COMVariant /*long*/ _Y)
{
}
```

XBasic

```
function Move as v (X as N,Y as N)
end function
```

dBASE

```
function nativeObject_Move(X,Y)
return
```

event Paint (hDC as Long, left as Long, top as Long, right as Long, bottom as Long)

Fired when the control is painting.

Type	Description
hDC as Long	A long expression that indicates the handle of the drawing context. Each drawing API uses the handle to device context.
left as Long	A long expression that indicates the left margin.
top as Long	A long expression that indicates the top margin.
right as Long	A long expression that indicates the right margin.
bottom as Long	A long expression that indicates the bottom margin.

The Paint event is called when the control paints its content. The Paint event provides owner draw support for the control. The /NET assembly provides the Paint event with two parameters of Graphics and Rectangle types.

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

C#
```
private void Paint(object sender,int hDC,int left,int top,int right,int bottom)
{
}
```

VB
```
Private Sub Paint(ByVal sender As System.Object,ByVal hDC As Integer,ByVal left As
Integer,ByVal top As Integer,ByVal right As Integer,ByVal bottom As Integer)
Handles Paint
End Sub
```

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

C#
```
private void Paint(object sender, AxEXDIALOGLib._IDialogEvents_PaintEvent e)
{
}
```

C++
```
void OnPaint(long hDC,long left,long top,long right,long bottom)
{
}
```

C++ Builder

```
void __fastcall Paint(TObject *Sender,long hDC,long left,long top,long right,long bottom)
{
}
```

Delphi

```
procedure Paint(ASender: TObject; hDC : Integer;left : Integer;top : Integer;right :
Integer;bottom : Integer);
begin
end;
```

Delphi 8 (.NET only)

```
procedure Paint(sender: System.Object; e:
AxEXDIALOGLib._IDialogEvents_PaintEvent);
begin
end;
```

Powe…

```
begin event Paint(long hDC,long left,long top,long right,long bottom)
end event Paint
```

VB.NET

```
Private Sub Paint(ByVal sender As System.Object, ByVal e As
AxEXDIALOGLib._IDialogEvents_PaintEvent) Handles Paint
End Sub
```

VB6

```
Private Sub Paint(ByVal hDC As Long,ByVal left As Long,ByVal top As Long,ByVal
right As Long,ByVal bottom As Long)
End Sub
```

VBA

```
Private Sub Paint(ByVal hDC As Long,ByVal left As Long,ByVal top As Long,ByVal
right As Long,ByVal bottom As Long)
End Sub
```

VFP

```
LPARAMETERS hDC,left,top,right,bottom
```

Xbas…

```
PROCEDURE OnPaint(oDialog,hDC,left,top,right,bottom)
RETURN
```

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

Java…
```
<SCRIPT EVENT="Paint(hDC,left,top,right,bottom)" LANGUAGE="JScript">
</SCRIPT>
```

VBSc…
```
<SCRIPT LANGUAGE="VBScript">
Function Paint(hDC,left,top,right,bottom)
End Function
</SCRIPT>
```

Visual Data…
```
Procedure OnComPaint Integer llhDC Integer llleft Integer lltop Integer llright
Integer llbottom
   Forward Send OnComPaint llhDC llleft lltop llright llbottom
End_Procedure
```

Visual Objects
```
METHOD OCX_Paint(hDC,left,top,right,bottom) CLASS MainDialog
RETURN NIL
```

X++
```
void onEvent_Paint(int _hDC,int _left,int _top,int _right,int _bottom)
{
}
```

XBasic
```
function Paint as v (hDC as N,left as N,top as N,right as N,bottom as N)
end function
```

dBASE
```
function nativeObject_Paint(hDC,left,top,right,bottom)
return
```

For instance, the following sample draw a text:

```
Option Explicit
Private Type RECT
    Left As Long
    Top As Long
    Right As Long
    Bottom As Long
End Type
Private Declare Function DrawFrameControl Lib "user32" (ByVal hDC As Long, lpRect As
RECT, ByVal un1 As Long, ByVal un2 As Long) As Long
```

```
Private Sub Dialog1_Paint(ByVal hDC As Long, ByVal Left As Long, ByVal Top As Long,
ByVal Right As Long, ByVal Bottom As Long)
   Dim rt As RECT
      rt.Left = Left
      rt.Right = Right
      rt.Top = Top
      rt.Bottom = Bottom
   DrawFrameControl hDC, rt, 0, 0
End Sub
```

event PictureChanged ()

Notifies your application once the control's Picture property is changed.

Type	Description

The PictureChanged event occurs once the control's Picture property is changed. The Picture property defines the shape of the control.

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

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

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

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

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

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

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

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

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

Powe...
```
begin event PictureChanged()
end event PictureChanged
```

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

VB6
```
Private Sub PictureChanged()
End Sub
```

VBA
```
Private Sub PictureChanged()
End Sub
```

VFP
```
LPARAMETERS nop
```

Xbas...
```
PROCEDURE OnPictureChanged(oDialog)
RETURN
```

Syntax for PictureChanged event, **/COM** version $^{(others)}$, on:

Java...
```
<SCRIPT EVENT="PictureChanged()" LANGUAGE="JScript">
</SCRIPT>
```

VBSc...
```
<SCRIPT LANGUAGE="VBScript">
Function PictureChanged()
End Function
</SCRIPT>
```

Visual Data...
```
Procedure OnComPictureChanged
    Forward Send OnComPictureChanged
End_Procedure
```

Visual Objects
```
METHOD OCX_PictureChanged() CLASS MainDialog
RETURN NIL
```

X++
```
void onEvent_PictureChanged()
{
}
```

XBasic

```
function PictureChanged as v ()
end function
```

dBASE

```
function nativeObject_PictureChanged()
return
```

event RClick ()

Occurs when the user right clicks the control.

Type	Description

Use the RClick event to notify your application that the user right clicks the control.

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

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

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

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

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

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

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

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

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

Powe…
```
begin event RClick()
```

```
end event RClick
```

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

VB6
```
Private Sub RClick()
End Sub
```

VBA
```
Private Sub RClick()
End Sub
```

VFP
```
LPARAMETERS nop
```

Xbas…
```
PROCEDURE OnRClick(oDialog)
RETURN
```

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

Java…
```
<SCRIPT EVENT="RClick()" LANGUAGE="JScript">
</SCRIPT>
```

VBSc…
```
<SCRIPT LANGUAGE="VBScript">
Function RClick()
End Function
</SCRIPT>
```

Visual Data…
```
Procedure OnComRClick
    Forward Send OnComRClick
End_Procedure
```

Visual Objects
```
METHOD OCX_RClick() CLASS MainDialog
RETURN NIL
```

X++
```
void onEvent_RClick()
{
}
```

XBasic

```
function RClick as v ()
end function
```

dBASE

```
function nativeObject_RClick()
return
```

event Show (ByRef Visible as Boolean)

Occurs whether the control is shown or hidden.

Type	Description
Visible as Boolean	(By Reference) A boolean expression that indicates whether the control is shown or hidden.

By default, the control is hidden when the container is deactivated. The Show event is fired each time when the control needs to change the visibility state. By handling the Show event you can control whether the control is visible or hidden no matter if the container is activated or deactivated. Use the Enabled property to show or hide the control. The Visible property of the control has no effect on the control.

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

C#
```
private void Show(object sender,ref bool Visible)
{
}
```

VB
```
Private Sub Show(ByVal sender As System.Object,ByRef Visible As Boolean)
Handles Show
End Sub
```

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

C#
```
private void Show(object sender, AxEXDIALOGLib._IDialogEvents_ShowEvent e)
{
}
```

C++
```
void OnShow(BOOL FAR* Visible)
{
}
```

C++ Builder
```
void __fastcall Show(TObject *Sender,VARIANT_BOOL * Visible)
{
}
```

Delphi
```
procedure Show(ASender: TObject; var Visible : WordBool);
begin
```

```
end;
```

Delphi 8 (.NET only)

```
procedure Show(sender: System.Object; e:
AxEXDIALOGLib._IDialogEvents_ShowEvent);
begin
end;
```

Powe…

```
begin event Show(boolean Visible)
end event Show
```

VB.NET

```
Private Sub Show(ByVal sender As System.Object, ByVal e As
AxEXDIALOGLib._IDialogEvents_ShowEvent) Handles Show
End Sub
```

VB6

```
Private Sub Show(Visible As Boolean)
End Sub
```

VBA

```
Private Sub Show(Visible As Boolean)
End Sub
```

VFP

```
LPARAMETERS Visible
```

Xbas…

```
PROCEDURE OnShow(oDialog,Visible)
RETURN
```

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

Java…

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

VBSc…

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

Visual Data…

```
Procedure OnComShow Boolean llVisible
   Forward Send OnComShow llVisible
```

```
End_Procedure
```

Visual Objects

```
METHOD OCX_Show(Visible) CLASS MainDialog
RETURN NIL
```

X++

```
void onEvent_Show(COMVariant /*bool*/ _Visible)
{
}
```

XBasic

```
function Show as v (Visible as L)
end function
```

dBASE

```
function nativeObject_Show(Visible)
return
```

For instance, the following sample lets the control visible no matter if the control is active or not-active:

```
Private Sub Dialog1_Show(Visible As Boolean)
    Visible = Dialog1.Enabled
End Sub
```