

# VSView 1.0 - VideoSoft Custom Control Library

To learn how to use help, press F1



## **Introduction**

Find out about [installation](#), [product support](#), and [licensing](#).



## **vsInForm**

Customize forms and controls, monitor the clipboard, drag and drop files from the File Manager.



## **vsPrinter**

Print files, text, graphics, and tables with automatic word wrap, headers and footers, multiple columns, and previewing.



## **vsViewPort**

Fit more controls on your windows with virtual scrollable areas.



## **vsDraw**

Create complex images, view them on the screen, copy them to the clipboard, and print them.

# Introduction

Welcome to VSView 1.0, another VideoSoft Custom Control Library.





VSView contains four custom controls designed to save you from writing tedious, repetitive, error-prone code. The controls are innovative and efficient. They are distributed as a single VBX to make installation easier.

Our distribution policy is almost as innovative as the controls. We want every Visual Basic programmer to get a copy of VSView and try it for as long as they want. Those who like the product and find it useful (almost everybody, we hope) can buy a license for a reasonable price. The only restriction is that unlicensed copies of VSView display a VideoSoft banner whenever they are loaded, to remind developers to license the product.

We hope you'll like VSView. If you have suggestions and ideas for new features or new controls, call us or write.

**VideoSoft**  
**2625 Alcatraz Avenue, Suite 271**  
**Berkeley, CA 94705**  
**(510) 547-7295 (phone)**  
**(510) 547-1084 (fax)**

## Control Summary

Icon	Object	Name	Description
	<u><b>vsInForm</b></u>	InForm	A control that you can drop into any container to customize its title bar, frame, resizing behavior, and frame buttons. InForm also allows you to monitor the clipboard, drag and drop files from File manager, and more.
	<u><b>vsPrinter</b></u>	Printer	A much improved printer object with word wrap, headers and footers, multi-column printing, graphics, and multi-page Print Preview capability.
	<u><b>vsViewPort</b></u>	ViewPort	A control that gives you a scrollable virtual area so you can fit more controls in your windows. Great for implementing Print Preview and programs that look like the Program Manager.
	<u><b>vsDraw</b></u>	Draw	A versatile drawing control that lets you create complex images, view them on the screen, copy them to the clipboard, or print them. Great for technical drawings, maps, and diagrams.

## Installation

To install VSView, just copy the following files to your WINDOWS\SYSTEM directory:

VSView.VBX	This file contains the controls. To use VSView from Visual Basic, you must include this file in your project.
VSView.LIC	This is the <b>run-time</b> license file. If VSView cannot find this file when called from a compiled VB application, it displays a VideoSoft banner and waits for the user to click Ok.
VSView.DEV	This is the <b>developers</b> license file. If VSView cannot find this file when loaded into Visual Basic, it displays a VideoSoft banner and waits for the user to click Ok.
VSView.HLP	This file contains the VSView on-line help.

If you would rather install VSView in a different directory, that's fine. As long as the help and license files are either in the same directory as the VBX, in the WINDOWS\SYSTEM directory, or in the WINDOWS directory, VSView will find them for you.

## Distribution

VSView is royalty-free. You may include copies of the VBX and LIC files with as many copies of as many applications you ship.

**You cannot distribute the developers license file VSView.DEV.**

If you work with other developers, you may be interested in VideoSofts site licenses. Call us for details.

# Product Support

Product support for VSView is available to licensed users through the following channels:

**CompuServe:** CIS 71552,3052

**Mail:** VideoSoft  
2625 Alcatraz Avenue, Suite 271  
Berkeley, California 94705

**Phone:** (510) 547-7295

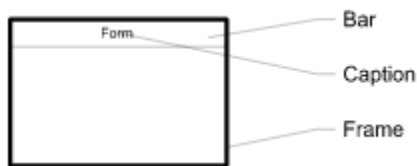
**Fax:** (510) 547-1084

Before calling for technical support, please make sure you know what version of VSView you are using. The version number appears in the About box that pops up when you double-click the About property in any of the VSView controls.



## InForm Reference

<b>Description</b>	<p>InForm allows you to customize the non-client area of its parent form or control. With InForm, you can add a chiseled look to your window borders and caption, add custom buttons, and control window resizing. And because InForm works with any container control, you can add these effects not only to forms, but also to Picture Boxes, Frames, Elastics, IndexTabs, and so on.</p> <p>InForm also allows you to monitor the clipboard, implement file drag-and-drop, and retrieve information about the environment.</p>
<b>File Name</b>	VSView.VBX
<b>Object Type</b>	vsInForm
<b>Remarks</b>	<p>InForm has many properties that are used to customize its parents non-client area (title bar and window frame.) These properties are divided in three groups: Bar... properties control the title bar, Cap... properties control the caption inside the title bar, and Frame... properties control the window frame. The picture below illustrates what parts of the window these names refer to:</p>



The CustomFrame property determines whether the InForm control should or should not take over managing the non-client area of the InForms parent. If you set this property to True and the parent is a form, the form must not have a menu. Remember also to set the forms ControlBox property to False, or users will be able to use alt-space to show the system menu.

When using InForm, keep in mind that one of Windows great virtues is that it promotes a standard interface across applications. Generally, it is not a good idea to change this standard. There are many exceptions to this rule, though: Visual Basic itself is a good example.

VBs floating toolbox and palette windows have skinny title bars, to save room. VBs main window can be resized horizontally, but not vertically. With InForm, you can do the same.

---

## InForm Summary

### Properties (default: Caption)

---

(About)	* <a href="#">AcceptFiles</a>	* <a href="#">BarColor</a>
* <a href="#">BarColorInactive</a>	* <a href="#">BarHeight</a>	* <a href="#">BarStyle</a>
* <a href="#">ButtonsLeft</a>	* <a href="#">ButtonsRight</a>	* <a href="#">CapAlign</a>
* <a href="#">CapColor</a>	* <a href="#">CapColorInactive</a>	* <a href="#">CapMultiLine</a>
* <a href="#">CapStyle</a>	Caption	* <a href="#">ClipMon</a>
* <a href="#">CustomFrame</a>	* <a href="#">FileName</a>	FontBold
FontItalic	FontName	FontSize
FontStrike	FontUnder	* <a href="#">FrameColor</a>
* <a href="#">FrameColorInactive</a>	* <a href="#">FrameCorners</a>	* <a href="#">FrameSizing</a>
* <a href="#">FrameStyle</a>	* <a href="#">FrameWidth</a>	* <a href="#">FreeGDI</a>
* <a href="#">FreeMemory</a>	* <a href="#">FreeSystem</a>	* <a href="#">FreeUser</a>
Hwnd	Index	Left
* <a href="#">MaxHeight</a>	* <a href="#">MaxWidth</a>	* <a href="#">MinHeight</a>
* <a href="#">MinWidth</a>	Name	* <a href="#">NumFiles</a>
* <a href="#">OnTop</a>	Parent	* <a href="#">PictLeft0</a>
* <a href="#">PictLeft1</a>	* <a href="#">PictLeft2</a>	* <a href="#">PictRight0</a>
* <a href="#">PictRight1</a>	* <a href="#">PictRight2</a>	Tag
Top	* <a href="#">Version</a>	

### Events

---

* <a href="#">ClickLButton</a>	* <a href="#">ClickRButton</a>	* <a href="#">DbClickLButton</a>
* <a href="#">DbClickRButton</a>	* <a href="#">DbClickCaption</a>	* <a href="#">DropFile</a>
* <a href="#">Move</a>	* <a href="#">NewClipboardData</a>	* <a href="#">Resize</a>

---

## AcceptFiles Property

<b>Description</b>	This property determines whether the user should be able to drop files from the File Manager into the Informs parent window..
<b>Usage</b>	<code>[form.]vsInForm.AcceptFiles [ = {True False} ]</code>
<b>Remarks</b>	If this property is set to True, the <a href="#">DropFile event</a> is fired when the user drops a file into the Informs parent.
<b>Default Value</b>	False
<b>Data Type</b>	Boolean

---

## BarColor, BarColorInactive Properties

<b>Description</b>	These are the colors used to paint the caption bar of the InForms parent. At run time, InForm chooses whether to use BarColor or BarColorInactive automatically, based on whether the InForms parent is active or not.
<b>Usage</b>	[ <i>form.</i> ]vslInForm. <b>BarColor</b> [ = <i>colorref</i> & ]
<b>Remarks</b>	<p>These properties are used only if the <u>CustomFrame property</u> is set to True.</p> <p>Setting either of these properties to zero causes the system default colors to be used. If you want the caption to be black, set these properties to one (if you simply pick black from the color box, VB will pick color zero and the control will use the default system colors).</p>
<b>Default Value</b>	Zero (use system colors)
<b>Data Type</b>	Color (Long)



---

## BarHeight Property

<b>Description</b>	This property sets or returns the height of the caption bar of the InForms parent, in pixels.
<b>Usage</b>	<code>[form.]vsInForm.<b>BarHeight</b> [ = <i>height%</i> ]</code>
<b>Remarks</b>	<p>This property is used only if the <u>CustomFrame property</u> is set to True.</p> <p>Setting this property to zero causes the system default caption height to be used. If you want a window with no caption, set the <u>BarStyle property</u> to None.</p>
<b>Default Value</b>	Zero (use system value)
<b>Data Type</b>	Integer

---

## BarStyle Property

**Description** This property determines the appearance of the caption bar of the InForms parent.

**Usage** `[form.]vsInForm.BarStyle [ = setting% ]`

**Remarks** Valid settings for this property are:

- 0 - None
- 1 - Classic
- 2 - NoBorder
- 3 - Raised
- 4 - Inset

This property is used only if the CustomFrame property is set to True.

**Default Value** 1 - Classic

**Data Type** Integer

---

## ButtonsLeft, ButtonsRight Properties

<b>Description</b>	These properties determine how many buttons should be displayed on the left and right-hand side of the InForm parents caption bar. The buttons are similar to the standard Windows maximize and minimize buttons, except you can customize the pictures in the buttons and attach custom code to clicks and double clicks.
<b>Usage</b>	[ <i>form.</i> ]vsInForm. <b>ButtonsLeft</b> [ = <i>setting%</i> ]
<b>Remarks</b>	<p>The number of buttons ranges from 0 to 3 on each side.</p> <p>These properties are used only if the <u>CustomFrame property</u> is set to True.</p>
<b>Default Value</b>	Zero
<b>Data Type</b>	Integer

---

## CapAlign Property

**Description** This property sets and returns the alignment of the text in the InForm parents caption bar.

**Usage** `[form.]vsInForm.CapAlign [ = setting% ]`

**Remarks** Valid settings for this property are:

0 - Left Top  
1 - Left Center  
2 - Left Bottom  
3 - Center Top  
4 - Center Center  
5 - Center Bottom  
6 - Right Top  
7 - Right Center  
8 - Right Bottom

This property is used only if the CustomFrame property is set to True.

**Default Value** 4 - Center Center

**Data Type** Integer

---

## CapColor, CapColorInactive Properties

<b>Description</b>	These are the colors used to paint the text in the caption bar of the InForms parent. At run time, InForm chooses whether to use CapColor or CapColorInactive automatically, based on whether the InForms parent is active or not.
<b>Usage</b>	[ <i>form.</i> ]vslInForm. <b>CapColor</b> [ = <i>colorref</i> & ]
<b>Remarks</b>	<p>These properties are used only if the <u>CustomFrame property</u> is set to True.</p> <p>Setting either of these properties to zero causes the system default colors to be used. If you want the caption to be black, set these properties to one (if you simply pick black from the color box, VB will pick color zero and the control will use the default system colors).</p>
<b>Default Value</b>	Zero (use system colors)
<b>Data Type</b>	Color (Long)

---

## CapMultiLine Property

<b>Description</b>	This property determines whether the the text in the InForm parents caption bar should wrap, allowing for multi-line captions.
<b>Usage</b>	<code>[form.]vsInForm.CapMultiLine [ = {True False} ]</code>
<b>Remarks</b>	This property is used only if the <u>CustomFrame property</u> is set to True.
<b>Default Value</b>	False
<b>Data Type</b>	Boolean

---

## CapStyle Property

**Description** This property determines the appearance of the text in the caption bar of the InForms parent.

**Usage** `[form.]vsInForm.CapStyle [ = setting% ]`

**Remarks** Valid settings for this property are:

0 - Flat  
1 - Raised  
2 - Inset

This property is used only if the CustomFrame property is set to True.

**Default Value** 0 - Flat

**Data Type** Integer

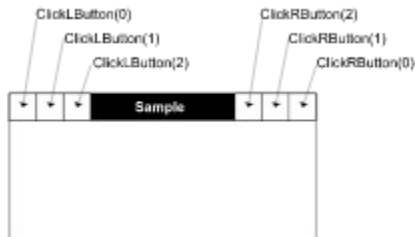
---

## ClickLButton, ClickRButton Events

**Description** Fired after one of the custom caption buttons is clicked.

**Syntax** **Sub** *vsInForm\_ClickLButton* (**Button** as Integer)

**Remarks** Buttons are numbered from the outside towards the center of the form, as shown in the diagram below:





---

## ClipMon Property

<b>Description</b>	Determines whether the InForm should monitor Clipboard activity.
<b>Usage</b>	<code>[form.]vsInForm.ClipMon [ = {True False} ]</code>
<b>Remarks</b>	If this property is set to True, the InForm fires the <u>NewClipboardData event</u> whenever the contents of the Clipboard change.
<b>Default Value</b>	False
<b>Data Type</b>	Boolean

---

## CustomFrame Property

<b>Description</b>	Determines whether the InForm should take over painting and managing the non-client area of its parent.
<b>Usage</b>	<code>[form.]vsInForm.CustomFrame [ = {True False} ]</code>
<b>Remarks</b>	If this property is set to true, the appearance of the InForms parent and its caption are controlled through the other InForm properties.
<b>Default Value</b>	False
<b>Data Type</b>	Boolean

---

## DblClickCaption Event

<b>Description</b>	Fired after the caption of the InForms parent is double-clicked.
<b>Syntax</b>	<b>Sub</b> <i>vsInForm_DblClickCaption</i> ()
<b>Remarks</b>	Normally, you should trap this event to implement the Windows standard behavior: flip the window state between maximized and normal.

---

## DbClickLButton, DbClickRButton Events

<b>Description</b>	Fired after one of the custom caption buttons is double-clicked.
<b>Syntax</b>	<b>Sub</b> <i>vsInForm_DbClickLButton</i> ( <b>Button as Integer</b> )
<b>Remarks</b>	Buttons are numbered from the outside towards the center of the form. See diagram in the description of the ClickLButton event.

---

## DropFile Event

<b>Description</b>	Fired when the user drops one or more files from another application such as the Program Manager or the File Manager into the InForms parent.
<b>Syntax</b>	<b>Sub</b> <i>vsInForm_DropFile</i> ()
<b>Remarks</b>	<p>To enable this feature, you must set the InForms <u><a href="#">AcceptFiles property</a></u> to True.</p> <p>To find out which files were dropped into the control, use the InForms NumFiles and FileName properties.</p>
<b>Example</b>	<pre>Sub vsInForm_DropFile ()     Dim i%     For i = 0 to vsInForm.NumFiles         ProcessFile vsInForm.FileName (i)     Next End Sub</pre>

---

## FileName Property

<b>Description</b>	This property returns the names of files dropped into the InForms parent.
<b>Usage</b>	<i>filename\$ = [form.]vsInForm.FileName(index%)</i>
<b>Remarks</b>	<p>This property is read-only and should only be used while handling the InForms DropFile event. The number of files dropped can be retrieved by reading the InForms <a href="#">NumFiles property</a>.</p> <p>See the description of the DropFile event for an example.</p>
<b>Data Type</b>	String Array

---

## FrameColor, FrameColorInactive Properties

<b>Description</b>	These are the colors used to paint the frame of the InForms parent. At run time, InForm chooses whether to use FrameColor or FrameColorInactive automatically, based on whether the InForms parent is active or not.
<b>Usage</b>	<code>[form.]vslInForm.FrameColor [ = colorref&amp; ]</code>
<b>Remarks</b>	<p>These properties are used only if the <u>CustomFrame property</u> is set to True.</p> <p>Setting either of these properties to zero causes the system default colors to be used. If you want the frame to be black, set these properties to one (if you simply pick black from the color box, VB will pick color zero and the control will use the default system colors).</p>
<b>Default Value</b>	Zero (use system colors)
<b>Data Type</b>	Color (Long)

---

## FrameCorners Property

<b>Description</b>	This property determines whether the InForm should mark the resizing corners on its parents frame.
<b>Usage</b>	<code>[form.]vslnForm.FrameCorners [ = {True False} ]</code>
<b>Remarks</b>	<p>This property is used only if the <u>CustomFrame property</u> is set to True.</p> <p>This property only affects the window appearance, not its resizing behavior.</p>
<b>Default Value</b>	True
<b>Data Type</b>	Boolean



---

## FrameSizing Property

<b>Description</b>	This property determines whether the InForm should allow its parent to be resized in both directions, in one direction, or not at all.
<b>Usage</b>	<code>[form.]vsInForm.FrameSizing [ = setting% ]</code>
<b>Remarks</b>	<p>Valid settings for this property are:</p> <ul style="list-style-type: none"><li>0 - None</li><li>1 - Horizontal</li><li>2 - Vertical</li><li>3 - Both</li></ul> <p>This property is used only if the <u>CustomFrame property</u> is set to True.</p>
<b>Default Value</b>	3 - Both
<b>Data Type</b>	Integer

---

## FrameStyle Property

<b>Description</b>	This property determines the appearance of the resizing frame around the InForms parent.
<b>Usage</b>	<code>[form.]vsInForm.FrameStyle [ = <i>setting</i>% ]</code>
<b>Remarks</b>	<p>Valid settings for this property are:</p> <ul style="list-style-type: none"><li>0 - None</li><li>1 - Classic</li><li>2 - Outside</li><li>3 - Raised Form</li><li>4 - Raised Frame</li></ul> <p>This property is used only if the <u>CustomFrame property</u> is set to True.</p>
<b>Default Value</b>	1 - Classic
<b>Data Type</b>	Integer

---

## FrameWidth Property

<b>Description</b>	This property sets or returns the width of the frame around the InForms parent, in pixels.
<b>Usage</b>	<code>[form.]vsInForm.FrameWidth [ = width% ]</code>
<b>Remarks</b>	<p>This property is used only if the <u>CustomFrame property</u> is set to True.</p> <p>Setting this property to zero causes the system default frame width to be used. If you want a window with no frame, set the FrameStyle property to None.</p>
<b>Default Value</b>	Zero (use system value)
<b>Data Type</b>	Integer

---

## FreeGDI Property

<b>Description</b>	This property returns the percentage of free Windows GDI resources.
<b>Usage</b>	<i>GDI%</i> = [ <i>form.</i> ] <i>vsInForm</i> . <b>FreeGDI</b>
<b>Remarks</b>	This property is read-only.
<b>Data Type</b>	Integer

---

## FreeMemory Property

<b>Description</b>	This property returns the amount of free memory available to Windows, in bytes.
<b>Usage</b>	<i>Mem&amp; = [form.]vsInForm.<b>FreeMemory</b></i>
<b>Remarks</b>	This property is read-only.
<b>Data Type</b>	Long

---

## FreeSystem Property

<b>Description</b>	This property returns the percentage of free Windows system resources.
<b>Usage</b>	<i>Sys%</i> = [ <i>form.</i> ] <i>vsInForm</i> . <b>FreeSystem</b>
<b>Remarks</b>	This property is read-only.
<b>Data Type</b>	Integer

---

## FreeUser Property

<b>Description</b>	This property returns the percentage of free Windows user resources.
<b>Usage</b>	<i>User%</i> = [ <i>form.</i> ]vsInForm. <b>FreeUser</b>
<b>Remarks</b>	This property is read-only.
<b>Data Type</b>	Integer

---

## MaxHeight, MinHeight Properties

<b>Description</b>	These properties allow you to set limits, in Twips, to the height of the InForms parent.
<b>Usage</b>	<code>[form.]vsInForm.MaxHeight [ = maxhei&amp; ]</code>
<b>Remarks</b>	<p>These properties are useful for windows that only show information along one direction, such as roll-up toolbars or VBs main window.</p> <p>These properties are used only if the <u>CustomFrame property</u> is set to True.</p>
<b>Default Value</b>	Zero
<b>Data Type</b>	Long



---

## MaxWidth, MinWidth Properties

<b>Description</b>	These properties allow you to set limits, in Twips, to the width of the InForms parent.
<b>Usage</b>	<code>[form.]vsInForm.MaxWidth [ = maxwid&amp; ]</code>
<b>Remarks</b>	<p>These properties are useful for windows that only show information along one direction, such as roll-up toolbars or VBs main window.</p> <p>These properties are used only if the <u>CustomFrame property</u> is set to True.</p>
<b>Default Value</b>	Zero
<b>Data Type</b>	Long

---

## Move Event

<b>Description</b>	Fired after the InForms parent has been moved.
<b>Syntax</b>	<b>Sub</b> <i>vsInForm_Move</i> <b>()</b>
<b>Remarks</b>	To enable this feature, you must set the InForms <u>ClipMon property</u> to True.

---

## NewClipboardData Event

<b>Description</b>	Fired whenever new data is copied to the Windows Clipboard. You can then use VBs Clipboard object to examine, retrieve, or change the clipboard contents.
<b>Syntax</b>	<b>Sub <i>vsInForm</i>_NewClipboardData ()</b>
<b>Remarks</b>	To enable this feature, you must set the InForms <u>ClipMon property</u> to True.

---

## NumFiles Property

<b>Description</b>	This property returns the number of files dropped into the InForms parent.
<b>Usage</b>	<i>numfiles%</i> = [ <i>form.</i> ]vsInForm. <b>NumFiles</b>
<b>Remarks</b>	<p>This property is read-only and should only be used while handling the InForms <u><a href="#">DropFile event</a></u>. The names of the files dropped can be retrieved by reading the InForms <u><a href="#">FileName property</a></u>.</p> <p>See the description of the <u><a href="#">DropFile event</a></u> for an example.</p>
<b>Data Type</b>	Integer

---

## OnTop Property

<b>Description</b>	Setting this property to True forces the InForms parent to remain on top of other windows, even when it is not active.
<b>Usage</b>	<code>[form.]vslnForm.OnTop [ = {True False} ]</code>
<b>Default Value</b>	False
<b>Remarks</b>	This is especially useful for floating toolbars and other small windows. A good way to implement OnTop functionality is to use one of the custom caption buttons to turn it on and off (see the example on page 6).
<b>Data Type</b>	Boolean

---

## PictLeft\*, PictRight\* Properties

<b>Description</b>	These properties are used to select the pictures for the InForm parents custom caption bar buttons. There are 6 properties in total, left and right ranging from 0 to 2.
<b>Usage</b>	<code>[form.]vsInForm.PictLeft0 [ = ImageControl ]</code>
<b>Remarks</b>	<p>The InForm draws the bevels automatically. All you have to supply is a small bitmap to be drawn on the button. For best results, make sure the background of the bitmaps is light gray, so it blends with the button background.</p> <p>These properties could have been implemented as picture arrays, but VB does not allow setting array properties at design-time.</p>
<b>Data Type</b>	Picture

---

## Resize Event

<b>Description</b>	Fired after the InForms parent has been resized.
<b>Syntax</b>	<b>Sub</b> <i>vsInForm_Move</i> ()
<b>Remarks</b>	<p>There is no need for this event if the InForms parent is a form, since forms have their own Resize event. However, you need this event to trap resizing of other controls.</p> <p>To enable this feature, you must set the InForms <u><a href="#">ClipMon property</a></u> to True.</p>

---

## Version Property

<b>Description</b>	This property returns the version of the InForm control currently loaded in memory.
<b>Usage</b>	<i>CheckVer%</i> = [ <i>form.</i> ]vsInForm. <b>Version</b>
<b>Remarks</b>	<p>You may want to check this value at the Form_Load event, to make sure the version that is executing is at least as current as the version used to develop your application.</p> <p>The version number is a three digit integer where the first digit represents the major version number and the last two represent the minor version number. For example, version 3.5 would return 350.</p> <p>This property is read-only.</p>
<b>Data Type</b>	Integer





## Printer Reference

<b>Description</b>	The VideoSoft Printer control allows you to create printed output quickly and easily. It supports multiple columns, text wrapping, headers and footers, and pictures. Best of all, it lets you do Print Previews just by setting a single property.
<b>File Name</b>	VSVIEW.VBX
<b>Object Type</b>	vsPrinter
<b>Remarks</b>	<p>The Printer control has a fixed coordinate system. It uses Twips as units, with the origin located at the top left of the page. To specify measurements in inches or points, remember that one inch is equivalent to 1440 Twips, and that one point is equivalent to 20 Twips.</p> <p>For convenience, two measurements are expressed in units other than Twips. Font sizes are expressed in points, and line spacing is expressed specified as a percentage of the current font height.</p> <p>When used in print preview mode, the scale is adjusted automatically so that a whole page always fits the control. This makes it easy to implement zoming in the print preview: all you have to do is resize the Print control, and its contents will stretch or shrink as needed.</p>

---

# Printer Summary

## Properties (default: Paragraph)

---

(Abort)	* <u>AbortCaption</u>	* <u>AbortWindow</u>
* <u>Action</u>	<u>BackColor</u>	<u>BorderStyle</u>
* <u>BrushColor</u>	* <u>BrushStyle</u>	* <u>Columns</u>
* <u>CurrentColumn</u>	* <u>CurrentLine</u>	* <u>CurrentPage</u>
* <u>CurrentX</u>	* <u>CurrentY</u>	* <u>Device</u>
* <u>Devices</u>	* <u>DPI</u>	* <u>Draw</u>
* <u>Driver</u>	<u>Enabled</u>	* <u>Error</u>
* <u>FileName</u>	<u>FontBold</u>	<u>FontItalic</u>
<u>FontName</u>	<u>FontSize</u>	<u>FontUnderline</u>
* <u>Footer</u>	* <u>Hdc</u>	* <u>HdrColor</u>
* <u>HdrFontBold</u>	* <u>HdrFontItalic</u>	* <u>HdrFontName</u>
* <u>HdrFontSize</u>	* <u>HdrFontUnderline</u>	* <u>Header</u>
<u>Height</u>	<u>HelpContextID</u>	* <u>IndentFirst</u>
* <u>IndentLeft</u>	* <u>IndentRight</u>	* <u>IndentTab</u>
<u>Index</u>	<u>Left</u>	* <u>LineSpacing</u>
* <u>MarginBottom</u>	* <u>MarginLeft</u>	* <u>MarginRight</u>
* <u>MarginTop</u>	* <u>Measure</u>	<u>MousePointer</u>
<u>Name</u>	* <u>NDevices</u>	* <u>Orientation</u>
* <u>PageBorder</u>	* <u>PageHeight</u>	* <u>PageWidth</u>
* <u>Paragraph</u>	<u>Parent</u>	* <u>PenColor</u>
* <u>PenStyle</u>	* <u>PenWidth</u>	* <u>Picture</u>
* <u>Polygon</u>	* <u>PolyLine</u>	* <u>Port</u>
* <u>Preview</u>	* <u>PreviewPage</u>	* <u>SpaceAfter</u>
* <u>Table</u>	* <u>TableBorder</u>	* <u>TableSep</u>
<u>Tag</u>	* <u>Text</u>	* <u>TextAngle</u>
* <u>TextAlign</u>	* <u>TextColor</u>	* <u>TextHei</u>
* <u>TextWid</u>	<u>Top</u>	* <u>Version</u>
<u>Visible</u>	<u>Width</u>	* <u>X1</u>
* <u>X2</u>	* <u>Y1</u>	* <u>Y2</u>

## Events

---

Click	DblClick	DragDrop
DragOver	* <u>EndDoc</u>	* <u>EndPage</u>
* <u>Error</u>	<u>KeyDown</u>	<u>KeyPress</u>
<u>KeyUp</u>	<u>MouseDown</u>	<u>MouseMove</u>
<u>MouseUp</u>	* <u>NewColumn</u>	* <u>NewLine</u>
* <u>NewPage</u>	* <u>NewTableCell</u>	* <u>StartDoc</u>

---

## AbortCaption Property

<b>Description</b>	Sets the caption for the default Abort Window.
<b>Usage</b>	[form.]vsPrinter. <b>AbortCaption</b> [ = caption\$ ]
<b>Remarks</b>	See the AbortWindow property for details on the default Abort Window.
<b>Default Value</b>	Printing...
<b>Data Type</b>	String

---

## AbortWindow Property

<b>Description</b>	Determines whether the Printer should automatically display an Abort Window while it prints.
<b>Usage</b>	<code>[form.]vsPrinter.<b>AbortWindow</b> [ = {<b>True</b> <b>False</b>} ]</code>
<b>Remarks</b>	<p>The automatic Abort Window shows the name of the document being printed (from the <u>FileName property</u>), the output device, port, and the current page. If the user presses the Abort button, printing stops and an <u>Error event</u> is fired.</p> <p>You may set the caption of the Abort Window with the AbortCaption property.</p> <p>Set this property to False if you want to display a custom Abort Window.</p>
<b>Default Value</b>	True
<b>Data Type</b>	Boolean

---

## Action Property

**Description** This property allows you to control the printer.

**Usage** `[form.]vsPrinter.Action = action%`

**Remarks** Valid settings for this property are:

- 0 - None
- 1 - Print File
- 2 - Choose Printer & Print File
- 3 - StartDoc
- 4 - NewPage
- 5 - NewCol
- 6 - EndDoc
- 7 - AbortDoc

The *Print File* action prints an entire text file using the current settings for font, margins, header and footer, and number of columns. The file name must be specified in advance by setting the FileName property. For example:

```
vsPrinter.FileName = C:\AUTOEXEC.BAT
vsPrinter.Action = 1 print the AUTOEXEC.BAT file
```

The *Choose Printer & Print File* action is similar to the *Print File* action, except it displays a dialog box that allows the user to choose the printer to be used.

The *StartDoc* and *EndDoc* properties are used to create and end a print job. All other printing commands should be called between these actions. The example below illustrates this:

```
vsPrinter.Action = 3 start document
If vsPrinter.Error Then always check for errors
Exit Sub
EndIf
vsPrinter = VideoSoft print some text
vsPrinter = 2625 Alcatraz Avenue, Suite 271
vsPrinter = Berkeley, CA 94705
vsPrinter = (510) 547-7295 (phone)
vsPrinter = (510) 547-1084 (fax)
vsPrinter.Action = 6 end document
```

The *NewPage* and *NewCol* actions can be used to force page and column breaks. If you use *NewCol* at the last column on a page, the current page is ejected and a fresh one is started.

The *AbortDoc* action cancels the current print job. This action should be used when you provide your own Abort window.

This property only acts at run-time, but the above list is also shown at design-time, on the properties window. This makes it easy for you to remember the code for each action.

This property is read-only.

**Data Type** Integer

---

## BrushColor Property

<b>Description</b>	Sets or returns the color to be used when filling objects.
<b>Usage</b>	<code>[form.]vsPrinter.<b>BrushColor</b> [ = colorref&amp; ]</code>
<b>Remarks</b>	This property affects the drawing of rectangles and ellipses (see the <a href="#">Draw property</a> ) as well as polygons (see the <a href="#">Polygon property</a> ).
<b>Default Value</b>	Black
<b>Data Type</b>	Long (Color)

---

## BrushStyle Property

<b>Description</b>	Sets or returns the style to be used when filling objects.
<b>Usage</b>	[ <i>form</i> .] <i>vsPrinter</i> . <b>BrushStyle</b> [ = <i>colorref</i> & ]
<b>Remarks</b>	<p>Valid settings for this property are:</p> <ul style="list-style-type: none"><li>0 - Solid</li><li>1 - Transparent</li><li>2 - Horizontal Line</li><li>3 - Vertical Line</li><li>4 - Upward Diagonal</li><li>5 - Downward Diagonal</li><li>6 - Cross</li><li>7 - Diagonal Cross</li></ul> <p>This property affects the drawing of rectangles and ellipses (see the <a href="#">Draw property</a>) as well as polygons (see the <a href="#">Polygon property</a>).</p>
<b>Default Value</b>	0 - Solid
<b>Data Type</b>	Long (Color)

---

## Columns Property

<b>Description</b>	Sets or returns the number of columns to be used when printing text.
<b>Usage</b>	<code>[form.]vsPrinter.<b>Columns</b> [ = numcolumns% ]</code>
<b>Remarks</b>	The setting of the Columns property affects the flow of text as you send it to the printer. The Print control does word wrapping, line, column, and page feeds automatically for you.
<b>Default Value</b>	1
<b>Data Type</b>	Integer

---

## CurrentColumn, CurrentLine, CurrentPage Properties

<b>Description</b>	These properties return the current page, column and line being printed by the Printer control.
<b>Usage</b>	<i>page%</i> = [ <i>form.</i> ]vsPrinter. <b>CurrentPage</b>
<b>Remarks</b>	These properties are read-only.
<b>Data Type</b>	Integer



---

## CurrentX, CurrentY Properties

<b>Description</b>	These properties are similar to the standard CurrentX and CurrentY properties available in VBs Picture control and Printer object.
<b>Usage</b>	<code>[form.]vsPrinter.<b>CurrentX</b> [ = <i>newpos</i>&amp; ]</code>
<b>Remarks</b>	As text is printed, these properties change to reflect the new cursor position. You may modify these properties directly if you want to set the cursor at a specific position. This may be necessary, for example, when centering text or printing tables.
<b>Data Type</b>	Long (Twips)

---

## DPI Property

<b>Description</b>	This property returns the resolution of the current printer, expressed as dots per inch.
<b>Usage</b>	<i>printres%</i> = [ <i>form.</i> ]vsPrinter. <b>DPI</b>
<b>Remarks</b>	This property is read-only.
<b>Data Type</b>	Integer

---

## Device Property

<b>Description</b>	This property sets or returns the name of the default Windows printer.
<b>Usage</b>	<code>[form.]vsPrinter.<b>Device</b> [ = devname\$ ]</code>
<b>Remarks</b>	<p>You can use this property to provide user feedback so they know which printer is being used or to set the default Windows printer.</p> <p>You cannot change this property while a job is printing. Trying to do so will trigger an Error event.</p> <p>You can only set this property to a valid device name. To obtain a list of valid device names, read the <a href="#">Devices</a> array property. Trying to set this property to an invalid device name will trigger an Error event.</p> <p>Changing this property will affect all Windows-based applications. Therefore, you may want to save the current printer before changing it and restore it later, as shown in the example below:</p> <pre><b>OldDevice\$ = vsPrinter1.Device</b>    <b>save original printer</b> vsPrinter.Device = vsPrinter.Devices(0)    select printer vsPrinter.Preview = False vsPrinter.Action = 3                    start job MyPrintingRoutine                      do printing vsPrinter.Action = 6                    finish job <b>vsPrinter1.Device = OldDevice\$</b>    <b>restore original printer</b></pre> <p>Changing this property automatically updates the Driver, Port, and DPI properties.</p>
<b>Data Type</b>	String

---

## Devices Property

<b>Description</b>	This property array contains a list of the printing devices available.
<b>Usage</b>	<code>devname\$ = [form.]vsPrinter.Devices(devindex%)</code>
<b>Remarks</b>	To find out how many devices are available, read the <u>NDevices property</u> .  This property is read only.
<b>Data Type</b>	String Array

---

## Draw Property

<b>Description</b>	This property allows you to draw lines, rectangles, and ellipses on the printer.
<b>Usage</b>	<code>[form.]vsPrinter.<b>Draw</b> = object%</code>
<b>Remarks</b>	<p>Valid settings for this property are:</p> <ul style="list-style-type: none"><li>0 - Nothing</li><li>1 - Line</li><li>2 - Rectangle</li><li>3 - Ellipse</li></ul> <p>All objects are drawn extending between the (x1, y1) and (x2, y2) points defined by the X1, Y1, X2, and Y2 properties.</p> <p>Objects are drawn with the current pen and filled with the current brush. Pen and brush attributes are defined with the Pen* and Brush* properties.</p> <p>This property only acts at run-time, but the above list is also shown at design-time, on the properties window. This makes it easy for you to remember the code for each object.</p> <p>This property is write-only.</p>
<b>Data Type</b>	Integer

---

## Driver Property

<b>Description</b>	This property returns the name of the current printer driver.
<b>Usage</b>	<i>drivename\$</i> = [ <i>form.</i> ]vsPrinter. <b>Driver</b>
<b>Remarks</b>	This property is read-only.
<b>Data Type</b>	String

---

## EndDoc Event

**Description**      This event gets fired after a document finishes printing.

**Syntax**            **Sub** *vsPrinter\_EndDoc* ()

---

## EndPage Event

**Description** This event is fired right before each page is ejected.

**Syntax** **Sub** *vsPrinter\_EndPage* ()



---

## Error Event

**Description** This event is fired when an error is detected while printing. The type of error can be determined by checking the Error property.

**Syntax** **Sub** *vsPrinter\_Error* ()

---

## Error Property

<b>Description</b>	This property is set whenever an error is detected while printing, or when the user aborts the printing process.
<b>Usage</b>	<code>errcode% = [form.]vsPrinter.<b>Error</b></code>
<b>Remarks</b>	<p>You should always check this property after starting a new print job and also while printing. The error code is reset whenever a new document is started.</p> <p>The error codes are:</p> <ul style="list-style-type: none"><li>0 - No Error</li><li>1 - Cant Open File</li><li>2 - Line Too Long</li><li>3 - Cant Access Printer</li><li>4 - Cant Start Job</li><li>5 - User Aborted</li><li>6 - Already Printing</li></ul> <p>The <i>Cant Open File</i> error occurs when you try to print a file using Action = 1 and the file specified by the <u>FileName property</u> cannot be opened.</p> <p>The <i>Line Too Long</i> error occurs when a text file is being printed but a line is too long to fit in memory. The line gets truncated.</p> <p>The <i>Cant Access Printer</i> and <i>Cant Start Job</i> errors occur when the printer is not available.</p> <p>The <i>User Aborted</i> error occurs when the users clicks the Abort button while a document is printing.</p> <p>The <i>Already Printing</i> error occurs when you try to start a new document while a document is being printed.</p> <p>This property is read-only</p>
<b>Default Value</b>	0 - No Error
<b>Data Type</b>	Integer

---

## FileName Property

<b>Description</b>	This property sets or retrieves the name of the text file being printed with the Action = 1 action. The file must exist, or an <u>Error event</u> is fired.
<b>Usage</b>	<code>[form.]vsPrinter.FileName [ = filename\$ ]</code>
<b>Remarks</b>	This property also sets the name of the current job, a string that appears in the default Abort Window and on the Print Manager list.
<b>Default Value</b>	(Empty String)
<b>Data Type</b>	String

---

## Footer Property

<b>Description</b>	This property sets or retrieves the text of the footer printed at the bottom of every page.
<b>Usage</b>	<code>[form.]vsPrinter.Footer [ = footer\$ ]</code>
<b>Remarks</b>	<p>The footer is composed of three sections, separated by pipe characters ( ). The first section is left-justified, the second is centered, and the third is right-justified.</p> <p>You may also include a page number field by embedding a %d code in the string. Do not use any percent signs in footers except for the page code.</p> <p>For example, the following footer would print the file name and page number on the left and right corners of every page:</p> <pre>vsPrinter.Footer = vsPrinter.FileName +   Page %d</pre> <p>The footer is printed using the font defined by the <u><a href="#">HdrFont* properties</a></u>.</p>
<b>Default Value</b>	(Empty String)
<b>Data Type</b>	String

---

## Hdc Property

<b>Description</b>	This property returns the Windows hdc (handle to device context) being used by the printer control.
<b>Usage</b>	<i>hdc%</i> = [ <i>form.</i> ]vsPrinter. <b>Hdc</b>
<b>Remarks</b>	<p>This property is useful if you wish to call Windows API functions to draw to the printer.</p> <p>The value of the Hdc property corresponds to a printer or metafile hdc, depending on whether the control is drawing to the printer or to the screen (in print preview mode).</p> <p>If theres no open document, the Hdc property returns zero.</p> <p>This property is read-only.</p>
<b>Data Type</b>	Integer

---

## HdrColor Property

<b>Description</b>	This property sets or retrieves the color of the font used to draw the document headers and footers.
<b>Usage</b>	<i>[form.]vsPrinter.HdrColor</i> [ = <i>colorref</i> & ]
<b>Default Value</b>	0 (Black)
<b>Data Type</b>	Long (Color)

---

## HdrFont\* Properties

**Description**      These properties are similar to the default Font\* properties, except they define the font used for printing the headers and footers, while the standard properties define the font used for printing regular text.

**Usage**            `[form.]vsPrinter.HdrFontName [ = fontname$ ]`  
`[form.]vsPrinter.HdrFontSize [ = size% ]`  
`[form.]vsPrinter.HdrFontBold [ = {True|False} ]`  
`[form.]vsPrinter.HdrFontItalic [ = {True|False} ]`  
`[form.]vsPrinter.HdrFontUnderline [ = {True|False} ]`

---

## Header Property

<b>Description</b>	This property sets or retrieves the text of the header printed at the top of every page.
<b>Usage</b>	<code>[form.]vsPrinter.Header [ = header\$ ]</code>
<b>Remarks</b>	<p>The header is composed of three sections, separated by pipe characters ( ). The first section is left-justified, the second is centered, and the third is right-justified.</p> <p>You may also include a page number field by embedding a %d code in the string. Do not use any percent signs in headers except for the page code.</p> <p>For example, the following header would print the file name and page number on the top left and right corners of every page:</p> <pre>vsPrinter.Header = vsPrinter.FileName +   Page %d</pre> <p>The header is printed using the font defined by the HdrFont* properties.</p>
<b>Default Value</b>	(Empty String)
<b>Data Type</b>	String



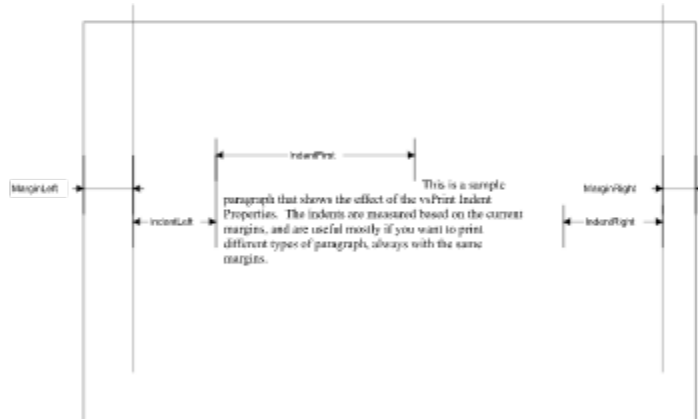
---

## IndentLeft, IndentRight, IndentFirst Properties

**Description** These properties set or retrieve the indentation to be used when printing paragraphs.

**Usage** `[form.]vsPrinter.IndentLeft [ = indent& ]`

**Remarks** The indents are measured from the current margins, as illustrated below:



**Default Value** 0

**Data Type** Long (Twips)

---

## IndentTab Property

<b>Description</b>	This property sets or retrieves the size of the tab stops used for printing text.
<b>Usage</b>	<code>[form.]vsPrinter.<b>IndentTab</b> [ = <i>indenttab</i> &amp; ]</code>
<b>Remarks</b>	Whenever a tab character is detected in the text, the Print control advances the print position to the next multiple of the IndentTab. If the next position is beyond the end of the current line, then the output advances to the next line.
<b>Default Value</b>	770 Twips (1/2)
<b>Data Type</b>	Long (Twips)

---

## LineSpacing Property

**Description** This property sets or retrieves the amount of space to leave between lines of text.

**Usage** `[form.]vsPrinter.LineSpacing [ = linespacing% ]`

**Remarks** Line spacing is expressed as a percentage of the current font height.

The following table summarizes common settings for the LineSpacing property:

Setting	Effect
100%	Single line spacing
150%	1.5 line spacing
200%	Double line spacing
50%	Half line spacing.

**Default Value** 100 (single line spacing)

**Data Type** Integer

---

## MarginBottom, MarginTop, MarginLeft, MarginRight Properties

<b>Description</b>	These properties set or retrieve the distance between the edge of the page and the printed text.
<b>Usage</b>	<code>[form.]vsPrinter.<b>MarginBottom</b> [ = <i>margin</i>&amp; ]</code>
<b>Default Value</b>	770 Twips (1/2)
<b>Data Type</b>	Long (Twips)

---

## Measure Property

<b>Description</b>	This property is used to measure the width and height of a string printed with the current font on the current device.
<b>Usage</b>	<code>[form.]vsPrinter.Measure [ = text\$ ]</code>
<b>Remarks</b>	To measure the text, assign it to this property. The width and height of the text are returned, in the TextWid and TextHei properties.  This property is write-only.
<b>Data Type</b>	String

---

## NDevices Property

<b>Description</b>	This property returns the number of printing devices currently installed.
<b>Usage</b>	<i>devices%</i> = [ <i>form.</i> ] <i>vsPrinter.NDevices</i>
<b>Remarks</b>	To access the name of a specific device, use the <code>Devices()</code> string array. This property is read-only.
<b>Data Type</b>	Integer

---

## NewColumn Event

**Description** This event is fired after each column break.

**Syntax** **Sub** *vsPrinter\_NewColumn* ()

---

## NewLine Event

**Description**      This event is fired after each line break.

**Syntax**            **Sub** *vsPrinter\_NewLine* ()



---

## NewPage Event

<b>Description</b>	This event is fired after each page is ejected.
<b>Syntax</b>	<b>Sub</b> <i>vsPrinter_</i> <b>NewPage ()</b>
<b>Remarks</b>	When this event is fired, the page is still blank. You can use this event to print custom headers and footers. You can even create watermarks, gray or transparent text and graphics that appear behind the document text.

---

## NewTableCell Event

<b>Description</b>	While printing a table with the <u><a href="#">Table property</a></u> , this event is fired before each cell is printed.
<b>Syntax</b>	<b>Sub vsPrinter_NewTableCell (Row As Integer, Column As Integer, Cell As String)</b>
<b>Remarks</b>	<p>This event allows you to customize the format of individual table cells.</p> <p>All parameters are read-only. The Row and Column parameters identify the cell about to be printed, and the Cell parameter contains the text that will be printed.</p> <p>For example, the following code customizes the third column of a table. It prints negative numbers in bold red and positive numbers in bold green:</p> <pre>Sub VSPrint_NewTableCell (Row%, Column%, Cell\$)     <b>** were only customizing column 3:</b>     If Column = 3 Then         VSPrinter1.FontBold = True         If Val (Cell\$) &lt; 0 Then             VSPrinter1.TextColor = RGB (255, 0, 0)         Else             VSPrinter1.TextColor = RGB (0, 255, 0)         <b>** restore regular formatting</b>     Else         VSPrinter1.FontBold = False         VSPrinter1.TextColor = 0     End If End Sub</pre>

---

## Orientation Property

<b>Description</b>	This property sets or retrieves the current paper orientation setting.
<b>Usage</b>	<code>[form.]vsPrinter.<b>Orientation</b> [ = <i>orient%</i> ]</code>
<b>Remarks</b>	Valid settings for this property are: 0 - Portrait 1 - Landscape
<b>Data Type</b>	Integer

---

## PageBorder Property

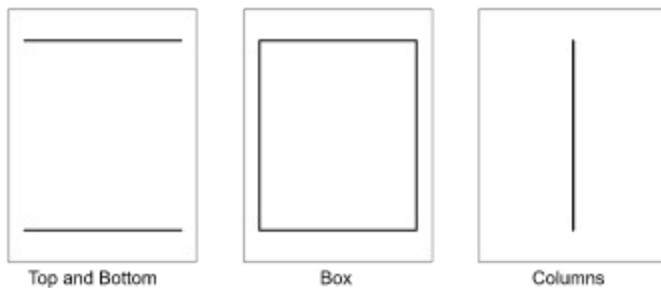
**Description** This property is used to set or retrieve the type of border to be drawn around each page.

**Usage** `[form.]vsPrinter.PageBorder [ = setting% ]`

**Remarks** Valid settings for this property are:

- 0 - None
- 1 - Bottom
- 2 - Top
- 3 - Top & Bottom
- 4 - Box
- 5 - Columns
- 6 - Columns, Top, & Bottom
- 7 - All

The effect of these settings is illustrated below:



The border is drawn using the pen defined by the Pen\* properties.  
See also the [Table property](#).

**Default Value** 3 - Top & Bottom

**Data Type** Integer

---

## PageHeight, PageWidth Properties

<b>Description</b>	These properties return the size of a page to be printed on the current printer, in Twips.
<b>Usage</b>	<i>pagehei%</i> = [ <i>form.</i> ]vsPrinter. <b>PageHeight</b>
<b>Remarks</b>	<p>These values are useful when drawing graphics on the page and also when implementing print preview. For example, to preview a printed page at 60% of its actual size, one could use the following code:</p> <pre>vsPrinter.Width = vsPrinter.PageWidth * 0.6 vsPrinter.Height = vsPrinter.PageHeight * 0.6</pre> <p>The above code would resize the Printer control to 60% of the size of a printed page, along with its contents.</p>
<b>Data Type</b>	Long (Twips)

---

## Paragraph Property

<b>Description</b>	This property is used to print a paragraph. The Printer control takes care of all the justification, word wrapping, and text flow.
<b>Usage</b>	<code>[form.]vsPrinter[.Paragraph] = text\$</code>
<b>Remarks</b>	After printing a string with this property, the Printer control will automatically skip a line and get ready for the next paragraph. If you want to print a paragraph in pieces, use the <u><a href="#">Text property</a></u> instead.  This property is write-only.
<b>Data Type</b>	String

---

## PenColor Property

<b>Description</b>	This property sets or returns the color of the pen used to draw all graphics and page borders.
<b>Usage</b>	<code>[form.]vsPrinter.<b>PenColor</b> [ = <i>colorref</i>&amp; ]</code>
<b>Remarks</b>	This property does not affect printed text. To change text color, use the TextColor and HdrColor properties.
<b>Default Value</b>	0 (Black)
<b>Data Type</b>	Long (Color)

---

## PenStyle Property

<b>Description</b>	This property sets or returns the style of the pen used to draw all graphics and page borders.
<b>Usage</b>	<code>[form.]vsPrinter.<b>PenStyle</b> [ = <i>setting%</i> ]</code>
<b>Remarks</b>	<p>Valid settings for this property are:</p> <ul style="list-style-type: none"><li>0 - Solid</li><li>1 - Dash</li><li>2 - Dot</li><li>3 - Dash-Dot</li><li>4 - Dash-Dot-Dot</li><li>5 - Transparent</li><li>6 - Inside Solid</li></ul>
<b>Default Value</b>	0 -Solid
<b>Data Type</b>	Integer



---

## PenWidth Property

<b>Description</b>	This property sets or returns the width of the pen used to draw all graphics and page borders.
<b>Usage</b>	<code>[form.]vsPrinter.<b>PenWidth</b> [ = width% ]</code>
<b>Remarks</b>	Setting PenWidth to zero causes the thinnest possible pen to be used. The pen width units are Twips.
<b>Default Value</b>	0
<b>Data Type</b>	Integer

---

## Picture Property

<b>Description</b>	This property allows you to print pictures on a page, at a specific position.
<b>Usage</b>	<code>[form.]vsPrinter.<b>Picture</b> = Image.Picture</code>
<b>Remarks</b>	<p>The position of the picture is determined by the X1, Y1, X2, and Y2 properties.</p> <p>The Printer control can handle bitmaps, icons, and metafiles in print and preview mode.</p> <p>Note that previewing large pictures requires a lot of memory. If you are going to print long documents with lots of large pictures, you may want to provide a draft preview mode which displays picture holders instead of actual pictures, as shown in the code below:</p> <pre>vsPrinter.Action = 3           start document vsPrinter.x1 = vsPrinter.MarginLeft + 1440  1 inch from left vsPrinter.y1 = vsPrinter.MarginTop + 1440  1 inch from top vsPrinter.x2 = vsPrinter.x1 + 2880        2 inches wide vsPrinter.y2 = vsPrinter.y1 + 2880        2 inches tall <b>If DraftPreview Then</b>     vsPrinter.Draw = 2           picture holder <b>Else</b>     vsPrinter.Picture = Picture1    picture <b>End If</b> vsPrinter.Action = 6           end document</pre> <p>This property is write-only.</p>
<b>Data Type</b>	Picture

---

## PolyLine Property

<b>Description</b>	This property allows you to plot a line composed of many segments.
<b>Usage</b>	<code>[form.]vsPrinter.PolyLine = points\$</code>
<b>Remarks</b>	<p>The <i>points\$</i> string contains a sequence of X, Y coordinates, in Twips, separated by spaces. Only the integer part of the coordinates is used. For example, the following code would draw a sine wave one inch tall across the page:</p> <pre>s = For i = 0 to 12 * PI Step PI/4     x = vsPrinter.PageWidth * i / (12 * PI)     y = vsPrinter.PageHeight / 2 + 770 * Sin(i)     s = s + Str(Int(x)) + Str(Int(y)) Next</pre> <p>You could create complex lines by repeatedly setting the <u>Draw property</u> to <i>Line</i> and updating X1, Y1, X2, and Y2, but the PolyLine is usually a more convenient and efficient way of obtaining the same results.</p> <p>The line is drawn with the current pen, defined by the Pen* properties.</p> <p>This property is write-only</p>
<b>Data Type</b>	String

---

## Polygon Property

<b>Description</b>	This property allows you to plot an arbitrary filled polygon. It is similar to the PolyLine property except that it produces a closed figure.
<b>Usage</b>	<code>[form.]vsPrinter.<b>Polygon</b> = <i>points</i>\$</code>
<b>Remarks</b>	For details on the syntax of the <i>points</i> \$ parameter, see the description of the PolyLine property.
<b>Data Type</b>	String

---

## Port Property

<b>Description</b>	This property returns the name of the port to which the current printer is connected.
<b>Usage</b>	<i>portname\$</i> = [ <i>form.</i> ]vsPrinter. <b>Port</b>
<b>Remarks</b>	This property is read-only.
<b>Data Type</b>	String

---

## Preview Property

**Description** This property determines whether Print control output should be sent to the printer or to the screen.

**Usage** [form.]vsPrinter.**Preview** [ = {True|False} ]

**Remarks** If the Preview property is set to False, the Print control sends its output to the printer. Unless the AbortWindow property is set to false, a default Abort window is created and shown to allow the user to abort the print job.

If the Preview property is set to True, the Print control saves all output in memory and allows you to display it, one page at a time, on the Print control itself. You can then switch pages by changing the PreviewPage property and scale the output by changing the size of the Print control.

This allows you to use the same code to create printed output and to perform print previews. The code below illustrates how this can be done:

**This routine sends output to the printer**

```
Sub PrintButton_Click ()  
    vsPrinter.Preview = False  
    DoPrinting  
End Sub
```

**This routines sends output to the Printer control**

```
Sub Preview_Click ()  
    vsPrinter.Preview = True  
    vsPrinter.PreviewPage = 1  
    DoPrinting  
End Sub
```

**This routine switches preview page**

```
Sub PreviewPage_Click (Index as Integer)  
    Select Case Index  
        Case 0:  
            vsPrinter.PreviewPage = 1                first  
        Case 1:  
            vsPrinter.PreviewPage = vsPrinter.PreviewPage + 1 next  
        Case 2:  
            vsPrinter.PreviewPage = vsPrinter.PreviewPage - 1 previous  
        Case 3:  
            vsPrinter.PreviewPage = vsPrinter.CurrentPage    last  
    End Select  
End Sub
```

In preview mode, the Printer control stretches the page image to fill the control. This makes it easy to implement print preview zooming, as illustrated by the code below:

**This routine zooms the preview page**

```
Sub ZoomPreviewPage (ZoomFactor as Integer)  
    vsPrinter.Width = vsPrinter.PageWidth * ZomFactor / 100  
    vsPrinter.Height = vsPrinter.PageHeight * ZomFactor / 100  
End Sub
```

The maximum number of pages in a preview document is 100.

**Default Value** False

**Data Type** Boolean

---

## PreviewPage Property

<b>Description</b>	This property sets or retrieves the number of the page being previewed.
<b>Usage</b>	<i>[form.]vsPrinter.PreviewPage</i> [ = <i>page%</i> ]
<b>Remarks</b>	Valid page numbers start at one and go up to the last page printed. The number of the last page can be retrieved by reading the CurrentPage property.
<b>Default Value</b>	1
<b>Data Type</b>	Integer

---

## SpaceAfter Property

<b>Description</b>	This property sets or retrieves the amount of vertical spacing between consecutive paragraphs. Units are Twips.
<b>Usage</b>	<code>[form.]vsPrinter.SpaceAfter [ = space&amp; ]</code>
<b>Remarks</b>	This space is automatically added after each paragraph when you use the <u><a href="#">Paragraph property</a></u> to print text.
<b>Default Value</b>	0
<b>Data Type</b>	Long (Twips)



---

## StartDoc Event

**Description**      This event gets fired before a document starts printing.

**Syntax**            **Sub** *vsPrinter\_StartDoc* ()

---

## Table Property

<b>Description</b>	This property is used to print tables. The Printer control takes care of all the justification, word wrapping, and text flow.
<b>Usage</b>	<code>[form.]vsPrinter.Table = table\$</code>
<b>Remarks</b>	<p>The table string is divided into rows and columns by special characters defined through the <a href="#">TableSep property</a>. By default, rows are separated by semi-colons (;) and columns by column pipes ( ), as shown in the example below.</p> <p>The first row contains formatting information only and is not printed. The formatting information consists of column alignment and width. The alignment is specified through the special characters &lt;, &gt;, and ^, for left, right, and center. The default is left alignment. The column widths are specified in Twips.</p> <p>The <a href="#">TableBorder</a> and Pen* properties determine the appearance of lines between the table entries, and you can customize the appearance of individual table cells by trapping the <a href="#">NewTableCell event</a>.</p> <p>For example, the following code draws a table with four rows and four columns. The columns are 0.5, 2, 1, and 2 inches wide:</p> <pre><b>** define format: first field is right-justified</b> fmt = &gt;770 2880 1440 2880; <b>** print table header in bold</b> tbl = fmt + Code Name Phone Address; vsPrinter.FontBold = True vsPrinter.Table = tbl <b>** print table body in normal</b> vsPrinter.FontBold = False tbl = fmt + 00123 John 415/324-2323 2312 Embarcadero South; tbl = tbl + 00432 Richard 303/321-2312 3212 Market; tbl = tbl + 02312 Susan 415/342-3421 666 Lombard; vsPrinter.Table = tbl</pre> <p>This property is write-only.</p>
<b>Data Type</b>	String

---

## TableBorder Property

<b>Description</b>	This property is used to set or retrieve the type of border to be used when drawing tables.
<b>Usage</b>	<code>[form.]vsPrinter.TableBorder [ = setting% ]</code>
<b>Remarks</b>	<p>Valid settings for this property are:</p> <ul style="list-style-type: none"><li>0 - None</li><li>1 - Bottom</li><li>2 - Top</li><li>3 - Top &amp; Bottom</li><li>4 - Box</li><li>5 - Columns</li><li>6 - Columns, Top, &amp; Bottom</li><li>7 - All</li></ul> <p>See also the <a href="#"><u>PageBorder property</u></a>.</p>
<b>Default Value</b>	7 - All
<b>Data Type</b>	Integer

---

## TableSep Property

<b>Description</b>	This property is used to specify the characters to be used as table row and column separators. It is used in conjunction with the <a href="#">Table property</a> .
<b>Usage</b>	<code>[form.]vsPrinter.<b>TableSep</b> = tablesep\$</code>
<b>Remarks</b>	<p>TheTableSep property must hold a two-character string. The first character is used as the column separator and the second as the row separator. The characters must be different.</p> <p>If the string is less than two characters long, or if the characters are equal, the default table separator string is used ( ;).</p>
<b>Default Value</b>	;
<b>Data Type</b>	String

---

## Text Property

<b>Description</b>	This property is used to print text. The Printer control takes care of all the justification, word wrapping, and text flow.
<b>Usage</b>	<code>[form.]vsPrinter.Text = text\$</code>
<b>Remarks</b>	<p>This property is used to print pieces of paragraphs. This is useful if you want to change fonts or colors halfway through a left-aligned paragraph. For example, the code below prints VB files and uses bold red type for function names:</p> <pre>Sub PrintVBFile (FileName as String)     Dim ln\$, subname\$, subargs\$      Open FileName For Input As #1     vsPrinter.Action = 3          <b>start document</b>     While Not EOF (1)         Line Input #1, ln          <b>print function header</b>         If Left(ln, 3) = Sub Then             ln = Mid(ln, 4)             subname = Left(ln, Instr (ln, ))             subargs = Mid(ln, Instr (ln, ))             vsPrinter.Text = Sub             vsPrinter.FontBold = True             vsPrinter.TextColor = RGB (255, 0, 0)             vsPrinter.Text = subname             vsPrinter.FontBold = False             vsPrinter.TextColor = 0             vsPrinter.Text = subargs              <b>print straight paragraph</b>         Else             vsPrinter = ln         End If     Wend      vsPrinter.Action = 6          <b>end document</b> End Sub</pre> <p>If you want to print an entire paragraph, use the <a href="#">Paragraph property</a> instead.</p> <p>This property is write-only.</p>
<b>Data Type</b>	String

---

## TextAlign Property

<b>Description</b>	This property sets the alignment of <u>paragraphs</u> and <u>tables</u> .
<b>Usage</b>	<code>[form.]vsPrinter.TextAlign = setting%</code>
<b>Remarks</b>	<p>Valid settings for this property are:</p> <ul style="list-style-type: none"><li>0 - Left</li><li>1 - Center</li><li>2 - Right</li></ul> <p>Tabs only work properly when text alignment is set to 0 (left).</p>
<b>Default Value</b>	0 - Left
<b>Data Type</b>	Integer

---

## TextAngle Property

<b>Description</b>	Specifies the angle, in tenths of degrees, between the base line of a character and the horizontal.
<b>Usage</b>	<code>[form.]vsDraw.<b>TextAngle</b> [ = <i>angle%</i> ]</code>
<b>Remarks</b>	The angle is measured in a counterclockwise direction.
<b>Default Value</b>	0
<b>Data Type</b>	Integer

---

## TextColor Property

<b>Description</b>	This property sets or retrieves the color used to print regular text.
<b>Usage</b>	<i>[form.]vsPrinter.TextColor</i> [ = <i>color</i> & ]
<b>Remarks</b>	Use the HdrColor to define the color used to print header and footer text.
<b>Default Value</b>	0 (Black)
<b>Data Type</b>	Long (Color)



---

## TextHei, TextWid Properties

<b>Description</b>	These properties are used to measure text, so you can do things such as draw boxes around strings and center text. They work in conjunction with the <a href="#">Measure property</a> .
<b>Usage</b>	<i>textwidth</i> & = [form.]vsPrinter. <b>TextHei</b>
<b>Remarks</b>	<p>To measure text, start by selecting the appropriate font through the Font* properties. Then, assign the text to be measured to the <a href="#">Measure property</a>. You can then read the text width and height, in Twips, from the TextHei and TextWid properties.</p> <p>The example below draws a string at the current cursor position with a box around it:</p> <pre>Sub BoxString (s As String)     vsPrinter.Measure = s     vsPrinter.X1 = vsPrinter.CurrentX     vsPrinter.Y1 = vsPrinter.CurrentY     vsPrinter.X2 = vsPrinter.CurrentX + vsPrinter.TextWid     vsPrinter.Y2 = vsPrinter.CurrentY + vsPrinter.TextHei     vsPrinter.Text = s           <b>draw text</b>     vsPrinter.Action = 2        <b>draw box around it</b> End Sub</pre> <p>These properties are read-only.</p>
<b>Data Type</b>	Long (Twips)

---

## Version Property

<b>Description</b>	This property returns the version of the Printer control currently loaded in memory.
<b>Usage</b>	<i>CheckVer%</i> = [ <i>form.</i> ]vsPrint. <b>Version</b>
<b>Remarks</b>	<p>You may want to check this value at the Form_Load event, to make sure the version that is executing is at least as current as the version used to develop your application.</p> <p>The version number is a three digit integer where the first digit represents the major version number and the last two represent the minor version number. For example, version 3.5 would return 350.</p> <p>This property is read-only.</p>
<b>Data Type</b>	Integer

---

## X1, X2, Y1, Y2 Properties

<b>Description</b>	These properties are used to control the graphics drawn using the Draw and Picture properties.
<b>Usage</b>	<code>[form.]vsPrinter.X1 [ = value&amp; ]</code>
<b>Remarks</b>	<p>These properties define a rectangle. Units are Twips, measured from the top left corner of the page.</p> <p>For an example of how these properties are used, see the description of the TextHei property.</p>
<b>Data Type</b>	Long (Twips)



## Draw Reference

<b>Description</b>	<p>The VideoSoft Draw control lets you create detailed, scalable, resolution-independent drawings. The drawings can be shown on your forms, printed, or copied to the Windows clipboard, where they become available for pasting into other Windows applications such as Word.</p> <p>After the drawing is created, you can change the drawing extents to provide distortion and zoom.</p> <p>Use the Draw control to create maps, charts, diagrams, or whatever graphics you need.</p>
<b>File Name</b>	VSView.VBX
<b>Object Type</b>	vsDraw
<b>Remarks</b>	<p>When you send drawing commands to the Draw control, they are stored in a list, but nothing is actually drawn on the screen or printer until you use the <u>Action property</u> to render the drawing. This is done this way to improve efficiency.</p> <p>The Draw control is based on Windows metafile technology, which has limited support for text manipulation. This makes it less than ideal for text-intensive applications. If you need to handle a lot of text, use the Printer control instead.</p>

---

## Draw Summary

### Properties (default: Action)

---

(About)	* <u>Action</u>	BackColor
* <u>BackStyle</u>	BorderStyle	* <u>BrushColor</u>
* <u>BrushStyle</u>	DragIcon	DragMode
* <u>Draw</u>	Enabled	FontBold
FontItalic	FontName	FontSize
FontStrike	FontUnder	Height
HelpContextID	Index	Left
Mousepointer	Name	Parent
* <u>PenColor</u>	* <u>PenStyle</u>	* <u>PenWidth</u>
* <u>Polygon</u>	* <u>PolyLine</u>	* <u>ScaleHeight</u>
* <u>ScaleLeft</u>	* <u>ScaleTop</u>	* <u>ScaleWidth</u>
Tag	* <u>TextAlign</u>	* <u>TextAngle</u>
* <u>TextColor</u>	* <u>TextOut</u>	Top
Visible	* <u>Version</u>	Width
* <u>X1</u>	* <u>X2</u>	* <u>Y1</u>
* <u>Y2</u>		

### Events

---

Click	DbIClick	DragDrop
DragOver	MouseDown	MouseMove
MouseUp		

### Methods

---

Move	Refresh	SetFocus
Zorder		

---

## Action Property

**Description** This property allows you to specify actions to be taken by the Draw control.

**Usage** `[form.]vsDraw.Action = action%`

**Remarks** Valid settings for this property are:

- 0 - None
- 1 - Clear
- 2 - Draw
- 3 - Print
- 4 - Choose Printer & Print
- 5 - Copy

The *Clear* action erases everything in the Draw control.

The *Draw* action renders the current drawing on the control. Note that you can create an entire drawing, but nothing will appear on the screen until you set the Action property to 2. This is done to improve rendering speed.

The *Print* action renders the drawing on the printer. The drawing is automatically scaled to fill as much of the page as possible, while preserving the aspect ratio of the drawing on the screen.

The *Choose Printer & Print* action is similar to the *Print* action, except it displays a dialog box that allows the user to choose the printer to be used.

The *Copy* action copies the current drawing to the Windows clipboard, where it becomes available to be pasted into other applications.

This property only acts at run-time, but the above list is also shown at design-time, on the properties window. This makes it easy for you to remember the code for each action.

This property is read-only.

### Data Type

Integer

---

## BackStyle Property

<b>Description</b>	This property determines whether text drawn on the Draw control is transparent or opaque.
<b>Usage</b>	<code>[form.]vsDraw.<b>BackStyle</b> [ = <i>setting</i>% ]</code>
<b>Remarks</b>	Valid settings for this property are: 0 - Transparent 1 - Opaque
<b>Default Value</b>	0 - Transparent
<b>Data Type</b>	Integer

---

## BrushColor Property

<b>Description</b>	Sets or returns the color to be used when filling objects.
<b>Usage</b>	<code>[form.]vsDraw.<b>BrushColor</b> [ = <i>colorref</i>&amp; ]</code>
<b>Remarks</b>	This property affects the drawing of rectangles and ellipses (see the <a href="#">Draw property</a> ) as well as polygons (see the <a href="#">Polygon property</a> ).
<b>Default Value</b>	Black
<b>Data Type</b>	Long (Color)



---

## BrushStyle Property

<b>Description</b>	Sets or returns the style to be used when filling objects.
<b>Usage</b>	<code>[form.]vsDraw.<b>BrushStyle</b> [ = <i>colorref</i>&amp; ]</code>
<b>Remarks</b>	<p>Valid settings for this property are:</p> <ul style="list-style-type: none"><li>0 - Solid</li><li>1 - Transparent</li><li>2 - Horizontal Line</li><li>3 - Vertical Line</li><li>4 - Upward Diagonal</li><li>5 - Downward Diagonal</li><li>6 - Cross</li><li>7 - Diagonal Cross</li></ul> <p>This property affects the drawing of rectangles and ellipses (see the Draw property) as well as polygons (see the <a href="#">Polygon property</a>).</p>
<b>Default Value</b>	0 - Solid
<b>Data Type</b>	Long (Color)

---

## Draw Property

<b>Description</b>	This property allows you to draw lines, rectangles, and ellipses.
<b>Usage</b>	<code>[form.]vsDraw.<b>Draw</b> = object%</code>
<b>Remarks</b>	<p>Valid settings for this property are:</p> <ul style="list-style-type: none"><li>0 - Nothing</li><li>1 - Line</li><li>2 - Rectangle</li><li>3 - Ellipse</li></ul> <p>All objects are drawn extending between the (x1, y1) and (x2, y2) points defined by the X1, Y1, X2, and Y2 properties.</p> <p>Objects are drawn with the current pen and filled with the current brush. Pen and brush attributes are defined with the Pen* and Brush* properties.</p> <p>This property only acts at run-time, but the above list is also shown at design-time, on the properties window. This makes it easy for you to remember the code for each object.</p> <p>This property is write-only.</p>
<b>Data Type</b>	Integer

---

## PenColor Property

<b>Description</b>	This property sets or returns the color of the pen used to draw all graphics.
<b>Usage</b>	<code>[form.]vsDraw.PenColor [ = colorref&amp; ]</code>
<b>Remarks</b>	This property does not affect text. To change text color, use the <a href="#">TextColor property</a> .
<b>Default Value</b>	0 (Black)
<b>Data Type</b>	Long (Color)

---

## PenStyle Property

<b>Description</b>	This property sets or returns the style of the pen used to draw all graphics.
<b>Usage</b>	<code>[form.]vsDraw.<b>PenStyle</b> [ = <i>setting%</i> ]</code>
<b>Remarks</b>	Valid settings for this property are: 0 - Solid 1 - Dash 2 - Dot 3 - Dash-Dot 4 - Dash-Dot-Dot 5 - Transparent 6 - Inside Solid
<b>Default Value</b>	0 -Solid
<b>Data Type</b>	Integer

---

## PenWidth Property

<b>Description</b>	This property sets or returns the width of the pen used to draw all graphics.
<b>Usage</b>	<code>[form.]vsDraw.<b>PenWidth</b> [ = <i>width</i>% ]</code>
<b>Remarks</b>	Setting PenWidth to zero causes the thinnest possible pen to be used.
<b>Default Value</b>	0
<b>Data Type</b>	Integer

---

## PolyLine Property

<b>Description</b>	This property allows you to plot a line composed of many points.
<b>Usage</b>	<code>[form.]vsDraw.PolyLine = points\$</code>
<b>Remarks</b>	<p>The <i>points\$</i> string contains a sequence of X, Y coordinates, separated by spaces. Only the integer part of the coordinates is used. The coordinates are arbitrary, determined by the Scale* properties.</p> <p>You could create complex lines by repeatedly setting the <u>Draw property</u> to <i>Line</i> and updating X1, Y1, X2, and Y2, but the PolyLine is usually a more convenient and efficient way of obtaining the same results.</p> <p>The line is drawn with the current pen, defined by the Pen* properties.</p> <p>This property is write-only</p>
<b>Data Type</b>	String

---

## Polygon Property

<b>Description</b>	This property allows you to plot an arbitrary filled polygon. It is similar to the PolyLine property except that it produces a closed figure.
<b>Usage</b>	<code>[form.]vsDraw.<b>Polygon</b> = <i>points</i>\$</code>
<b>Remarks</b>	For details on the syntax of the <i>points</i> \$ parameter, see the description of the PolyLine property.
<b>Data Type</b>	String

---

## ScaleHeight, ScaleWidth Properties

<b>Description</b>	These properties set or return the extents of the coordinate system used by the Draw control.
<b>Usage</b>	<code>[form.]vsDraw.ScaleHeight = height%</code>
<b>Remarks</b>	<p>All drawing on the Draw control is done in an arbitrary coordinate system, determined by the Scale* properties.</p> <p>For example, the following code draws a rectangle that fills the entire Draw control, regardless of its physical dimensions:</p> <pre>vsDraw.X1 = 0 vsDraw.Y1 = 0 vsDraw.X2 = 1000 vsDraw.Y2 = 1000 vsDraw.Draw = 2   <b>Draw = Rectangle</b> vsDraw.Action = 2 <b>Action = Draw</b></pre> <p>You can stretch drawings, zoom, or scroll by modifying the coordinate system and setting the <u>Action property</u> to 2 (Draw).</p>
<b>Default Value</b>	1000
<b>Data Type</b>	Integer



---

## ScaleLeft, ScaleTop Properties

<b>Description</b>	These properties set or return the logical origin of the coordinate system used by the Draw control.
<b>Usage</b>	<code>[form.]vsDraw.<b>ScaleLeft</b> = left%</code>
<b>Remarks</b>	<p>All drawing on the Draw control is done in an arbitrary coordinate system, determined by the Scale* properties.</p> <p>For example, the following code sets the (0,0) point of the logical coordinate system to the center of the Draw control, regardless of its physical dimensions:</p> <pre>vsDraw.ScaleLeft = -vsDraw.ScaleWidth / 2 vsDraw.ScaleTop = -vsDraw.ScaleHeight / 2</pre> <p>You can stretch drawings, zoom, or scroll by modifying the coordinate system and setting the <u>Action property</u> to 2 (Draw).</p>
<b>Default Value</b>	1000
<b>Data Type</b>	Integer

---

## TextAlign Property

<b>Description</b>	This property sets the alignment to be used when drawing text.
<b>Usage</b>	<code>[form.]vsDraw.TextAlign = <i>setting</i>%</code>
<b>Remarks</b>	Valid settings for this property are: 0 - Left 1 - Center 2 - Right
<b>Default Value</b>	0 - Left
<b>Data Type</b>	Integer

---

## TextAngle Property

<b>Description</b>	Specifies the angle, in tenths of degrees, between the base line of a character and the horizontal.
<b>Usage</b>	<code>[form.]vsDraw.<b>TextAngle</b> [ = <i>angle%</i> ]</code>
<b>Remarks</b>	The angle is measured in a counterclockwise direction when the y direction is down and in a clockwise direction when the y direction is up.
<b>Default Value</b>	0
<b>Data Type</b>	Integer

---

## TextColor Property

<b>Description</b>	This property sets or retrieves the color used to print text.
<b>Usage</b>	<code>[form.]vsDraw.<b>TextColor</b> [ = <i>color</i>&amp; ]</code>
<b>Default Value</b>	0 (Black)
<b>Data Type</b>	Long (Color)

---

## TextOut Property

<b>Description</b>	Assigning a string to this property causes the Draw control to print the string at the current cursor position, determined by the X1 and Y1 properties.
<b>Usage</b>	<code>[form.]vsDraw.TextOut = text\$</code>
<b>Remarks</b>	<p>The cursor position is not updated after printing.</p> <p>The string assigned to TextOut may have embedded carriage-returns (chr\$(13)) characters, which cause line breaks.</p> <p>This property is write-only.</p>
<b>Data Type</b>	String

---

## Version Property

<b>Description</b>	This property returns the version of the Draw control currently loaded in memory.
<b>Usage</b>	<i>CheckVer%</i> = [ <i>form.</i> ]vsDraw. <b>Version</b>
<b>Remarks</b>	<p>You may want to check this value at the Form_Load event, to make sure the version that is executing is at least as current as the version used to develop your application.</p> <p>The version number is a three digit integer where the first digit represents the major version number and the last two represent the minor version number. For example, version 3.5 would return 350.</p> <p>This property is read-only.</p>
<b>Data Type</b>	Integer

---

## X1, X2, Y1, Y2 Properties

<b>Description</b>	These properties are used to control the graphics drawn using the <u>Draw property</u> and text drawn with the <u>TextOut property</u> .
<b>Usage</b>	[ <i>form.</i> ]vsDraw.X1 [ = <i>value</i> % ]
<b>Remarks</b>	These properties define a rectangle. The coordinates are arbitrary, determined by the Scale* properties.
<b>Data Type</b>	Integer



## ViewPort Reference

<b>Description</b>	The VideoSoft ViewPort is a scrollable container control. With ViewPort, you no longer have to write tedious code to synchronize scroll bars and picture boxes. You decide how big your window should be, then let the ViewPort scroll your controls for you.
<b>File Name</b>	VSView.VBX
<b>Object Type</b>	vsViewPort
<b>Remarks</b>	<p>The ViewPort automatically scrolls all its windowed child controls, but it does not scroll graphical controls such as Labels and Image Boxes.</p> <p>Use the ViewPort with the VideoSoft Print control to implement print preview in your programs.</p>



---

## ViewPort Summary

### Properties (default: AutoScroll)

---

(About)	* <u>AutoScroll</u>	BackColor
BorderStyle	Height	HelpContextID
Hwnd	Index	* <u>LargeChangeHorz</u>
* <u>LargeChangeVert</u>	Left	Name
Parent	* <u>SmallChangeHorz</u>	* <u>SmallChangeVert</u>
Tag	Top	* <u>Track</u>
* <u>Version</u>	* <u>VirtualHeight</u>	* <u>VirtualLeft</u>
* <u>VirtualTop</u>	* <u>VirtualWidth</u>	Width

### Events

---

Click	DbClick	DragDrop
DragOver	KeyDown	KeyPress
KeyUp	MouseDown	MouseMove
MouseUp	* <u>Scroll</u>	

---

## AutoScroll Property

<b>Description</b>	This property determines whether ViewPort should automatically scroll its contents when the user clicks the scroll bars.
<b>Usage</b>	<code>[form.]vsViewPort.<b>AutoScroll</b> [ = {True False} ]</code>
<b>Remarks</b>	<p>This property should be set to True in most applications, so that scrolling is done automatically by the ViewPort control.</p> <p>If you wish to process scrolling yourself, set AutoScroll to False and respond to the ScrollEvent.</p>
<b>Default Value</b>	True
<b>Data Type</b>	Boolean

---

## LargeChangeHorz, LargeChangeVert Properties

<b>Description</b>	These properties determine the amount of change to the VirtualLeft and VirtualTop properties when the user clicks the area between the scroll box and scroll arrow. Units are Twips.
<b>Usage</b>	<code>[form.]vsViewPort.LargeChangeHorz [ = <i>setting</i> &amp; ]</code>
<b>Default Value</b>	300 Twips
<b>Data Type</b>	Long

---

## Scroll Event

<b>Description</b>	Fired whenever the values of VirtualLeft and VirtualTop change.
<b>Syntax</b>	<b>Sub</b> <i>vsViewPort_Scroll</i> ()
<b>Remarks</b>	If AutoScroll is set to True, the Scroll event is fired after the actual scrolling takes place. If AutoScroll is set to False, the Scroll event is fired anyway.

---

## SmallChangeHorz, SmallChangeVert Properties

<b>Description</b>	These properties determine the amount of change to the VirtualLeft and VirtualTop properties when the user clicks the scroll arrow. Units are Twips.
<b>Usage</b>	<code>[form.]vsViewPort.SmallChangeHorz [ = <i>setting</i> &amp; ]</code>
<b>Default Value</b>	30 Twips
<b>Data Type</b>	Long

---

## Track Property

<b>Description</b>	This property determines whether ViewPort should perform scrolling while the user moves the scroll boxes.
<b>Usage</b>	<code>[form.]vsViewPort.Track [ = {True False} ]</code>
<b>Remarks</b>	Setting Track to true provides more user feedback during scrolling, but it can slow down your application and cause some flicker. If in doubts, try it both ways and see which works best for you.
<b>Default Value</b>	False
<b>Data Type</b>	Boolean

---

## VirtualHeight, VirtualWidth Properties

<b>Description</b>	These properties determine the extent of the area that can be seen by scrolling, regardless of the actual size of the control. Units are Twips.
<b>Usage</b>	<code>[form.]vsViewPort.VirtualHeight [ = <i>setting</i> &amp; ]</code>
<b>Remarks</b>	<p>If the VirtualHeight is smaller than the actual height of the control, the ViewPort automatically hides the vertical scroll bar. The same applies to VirtualWidth and the horizontal scroll bar.</p> <p>See also the description of the VirtualLeft and VirtualTop properties.</p>
<b>Default Value</b>	0 Twips
<b>Data Type</b>	Long

---

## Version Property

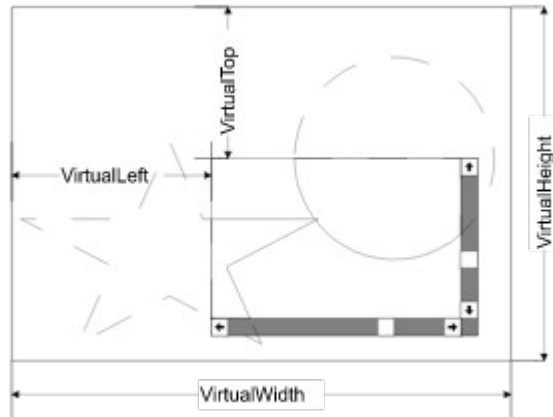
<b>Description</b>	This property returns the version of the ViewPort control currently loaded in memory.
<b>Usage</b>	<i>CheckVer%</i> = [ <i>form.</i> ]vsViewPort. <b>Version</b>
<b>Remarks</b>	<p>You may want to check this value at the Form_Load event, to make sure the version that is executing is at least as current as the version used to develop your application.</p> <p>The version number is a three digit integer where the first digit represents the major version number and the last two represent the minor version number. For example, version 3.5 would return 350.</p> <p>This property is read-only.</p>
<b>Data Type</b>	Integer



---

## VirtualLeft, VirtualTop Properties

<b>Description</b>	These properties determine what portion of the scrollable area is visible at any time. Units are Twips.
<b>Usage</b>	[ <i>form.</i> ]vsViewPort. <b>VirtualLeft</b> [ = <i>setting</i> & ]
<b>Remarks</b>	The diagram below illustrates the relationship between VirtualLeft/Top and VirtualWidth/Height:



If the AutoScroll property is set to True, then changes to these properties cause ViewPort to scroll its contents and to fire the Scroll event.

<b>Default Value</b>	0 Twips
<b>Data Type</b>	Long

