

**ActiveContainer Control v3.0**  
**by Galileo Technologies**  
**©1997, Galileo Technologies All Rights Reserved**

Final Version release: 3.00.0001

**[Click Here to go to the Documentation](#)**

---

-----

Thank you for your interest in the ActiveContainer ActiveX Control! It is only by support from users like you that we are able to remain motivated to produce such high-quality software as that which you are using now. This product is freeware, and as a result this help file is not as complete as we would like to see it, but since we aren't getting paid for this, we have left out some extraneous information. This help file outlines just the main documentation and help for the ActiveContainer Control, items such as examples and design specifications have been left out of this help file due to their time consumption and complexity. However, if you are unable to have your questions answered by this help file, please feel free to contact us at any time at 'galileotech@juno.com'. We typically have a support technician on hand 24-hours a day, so your question will be answered quickly and efficiently. All comments on this product are welcome, email us! Good luck with ActiveContainer, and happy coding!

-Benjamin

Benjamin Crane, CEO  
Galileo Technologies  
galileotech@juno.com  
4015 91st Ave SE  
Mercer Is., WA 98040

**Also, please read the [EULA\(End User License Agreement\)](#).  
[Old Documentation](#) is also available to look at.**

This help file was created with [HelpScribble](#).

# ActiveContainer

The ActiveContainer control is a control container that is able to be used to imitate the "docked forms" style in the VB5 IDE. The control is alignable and has a control box to allow it to be placed on an MDI form and be closeable. In addition, the RuntimeDockable and RuntimeResizable properties allow the ActiveContainer control to employ "Active Docking" (the docking behavior of forms in the VB5 IDE) and "Active Resizing" (the control may be resized at run time), respectively.

## Methods/Functions

- FadeBack
- PopUpMenu
- Refresh

## Properties

- BackColor
- BackStyle
- BorderColor
- Caption
- Enabled
- Font
- ForeColor
- Mouselcon
- MousePointer
- RunTimeDockable
- RunTimeResizable
- TitleBarColor

## Events

- Click
- MouseDown
- MouseMove
- MouseUp
- Paint
- XBoxClick

# FadeBack

**Type:** Function

**Return Type:** Long

**Syntax:** `h = controlName.FadeBack(Red As Integer, Green As Integer, Blue As Integer, [Optional Bands As Long])`

**Arguments:**

**Red** - Required, defines whether the fade algorithm incorporates the Red color. Treat as a Boolean data type, True or False.

**Green** - Required, defines whether the fade algorithm incorporates the Green color. Treat as a Boolean data type, True or False.

**Blue** - Required, defines whether the fade algorithm incorporates the Blue color. Treat as a Boolean data type, True or False.

**Bands** - Optional, defines the number of bands the fade algorithm uses to create the effect.

**Description:**

The FadeBack function allows the user to define a fading background color for the control selected. The arguments (Red, Green, Blue) define which colors will be incorporated in the fade effect. Treat the arguments as Boolean data types, so only use **True** or **False** as the argument settings, indicating whether or not you wish to incorporate that particular color. The Bands argument allows you to custom set the number of bands in the fade effect, the default is 256.

## PopUpMenu

**Type:** Method

**Syntax:** *controlName*.PopUpMenu(Menu As Object, [Flags], [X], [Y], [DefaultMenu])

**Arguments:**

**Menu** - Required, defines the Menu object to display.

**Flags** - Optional, defines flag stipulations.

**X** - Optional, defines the x-plane coordinate at which to display the menu.

**Y** - Optional, defines the y-plane coordinate at which to display the menu.

**DefaultMenu** - Optional, defines the default menu to display

**Description:**

Displays a pop-up menu on an object. Use the X and Y arguments to specifically tell the computer where to display the menu.

This help file was created with [HelpScribble](#).

# Refresh

**Type:** Method

**Syntax:** *controlname*.Refresh()

**Description:**

Forces a complete repaint of a object.

This help file was created with [HelpScribble](#).

# BackColor

**Type:** Property

**Data Type:** OLE\_COLOR

**Description:**

Returns/sets the background color used to display text and graphics in an object.

This help file was created with [HelpScribble](#).

# BackStyle

**Type:** Property

**DataType:** Integer

**Description:**

Indicates whether a Label or the background of a Shape is transparent or opaque.

This help file was created with [HelpScribble](#).

# BorderColor

**Type:** Property

**Data Type:** OLE\_COLOR

**Description:**

Returns/sets the color used to draw the borders of an object.

This help file was created with [HelpScribble](#).



# Caption

**Type:** Property

**Data Type:** String

**Description:**

Returns/sets the text displayed in the control's title bar or below the control's icon.

This help file was created with [HelpScribble](#).

# Enabled

**Type:** Property

**Data Type:** Boolean

**Description:**

Returns/sets a value that determines whether an object can respond to user-generated events.

This help file was created with [HelpScribble](#).

# Font

**Type:** Property

**Data Type:** StdFont

**Description:**

Returns a Font object. Used for setting the font of the Caption in the selected control.

This help file was created with [HelpScribble](#).

# ForeColor

**Type:** Property

**Data Type:** Long

**Description:**

Returns/sets the foreground color used to display text and graphics in an object.

This help file was created with [HelpScribble](#).

# Mouselcon

**Type:** Property

**DataType:** StdPicture

**Description:**

Sets a custom mouse icon.

This help file was created with [HelpScribble](#).

# MousePointer

**Type:** Property

**Data Type:** Integer

**Description:**

Returns/sets the type of mouse pointer displayed when over part of an object.

This help file was created with [HelpScribble](#).

# RunTimeDockable

**Type:** Property

**DataType:** Boolean

**Description:**

Sets whether or not the control employ's Active Docking. When TRUE, the control is able to be dynamically docked at run time.

This help file was created with [HelpScribble](#).

# RunTimeResizable

**Type:** Property

**DataType:** Boolean

**Description:**

Sets whether or not the control employ's Active Resizing. When TRUE, the control is able to be dynamically resized at run time.

This help file was created with [HelpScribble](#).



# TitleBarColor

**Type:** Property

**DataType:** OLE\_COLOR

**Description:**

Returns/Sets the color of the control's title bar.

This help file was created with [HelpScribble](#).

# Paint

Occurs when any part of the form or control is moved, enlarged, or exposed.

This help file was created with [HelpScribble](#).

# MouseDown

Occurs when the user presses the mouse button while an object has the focus.

This help file was created with [HelpScribble](#).

# MouseMove

Occurs when the user moves the mouse.

This help file was created with [HelpScribble](#).

# MouseUp

Occurs when the user releases the mouse button while an object has the focus.

This help file was created with [HelpScribble](#).

## XBoxClick

Occurs when the user presses and then releases a mouse button over the ActiveContainer's control box.

This help file was created with [HelpScribble](#).

# Click

Occurs when the user presses and then releases a mouse button over an object.

This help file was created with [HelpScribble](#).

## EULA (End User License Agreement)

**EULA (End User License Agreement):** The **User** of this **Software** (ActiveVB5 Tools OCX) is entitled to freely use and distribute the **Software** provided that it remains intact and in its original form. Additionally, the **User** must notify the **Customer** that this **Software** is copyrighted by Galileo Technologies. This **Software** may be used for educational, personal, or corporate applications provided that this **Software** is not sold or used to gain profit on its own. This **Software** is used "as is" and Galileo Technologies takes no responsibility for any damage, implicit or explicit, done to the **User's** or the **Customer's** computer system(s). Use of this **Software** implies agreement with these terms. If you do not agree to these terms, you may not use this **Software**.

This help file was created with [HelpScribble](#).



## Old Documentation

```
*****
***
GALILEO TECHNOLOGIES TAKES NO RESPONSIBILITY FOR ANY DAMAGE, IMPLICIT OR
EXPLICIT, MALFUNCTION OR CORRUPTION OF THE USER'S COMPUTER SYSTEM DONE BY THE
ACTIVEVB5TOOLS OCX PACKAGE.  THE USER MUST AGREE THAT THIS SOFTWARE IS "AS IS"
AND USES THE SOFTWARE AT HIS OR HER OWN RISK.  BY USE OF THIS SOFTWARE, THE USER
AGREES TO THESE TERMS.
*****
***
```

Alrighty, its time to start working with the ActiveVB5Tools OCX Package! This release (2.0 final) contains the following controls:

- vbActiveButton
- vbActiveContainer
- vbActiveMenu
- vbActiveSpacer
- vbActiveToolbar
- vbComponentWizard
- vbMenuButton

\*NOTE\* PLEASE DOWNLOAD THE LATEST README.TXT FILE FROM  
[HTTP://WWW.GEOCITIES.COM/PARIS/5341/README.TXT](http://www.geocities.com/paris/5341/README.TXT)  
IT WILL CONTAIN ALL THE NECESSARY INFORMATION ABOUT THIS LATEST RELEASE.

The Controls:

ActiveButton -

The ActiveButton control is a general-use activeborder button. The main useful properties are the Caption property (allows the user to set a caption for the button) and a Picture property (allows the user to place a picture on the backdrop of the control). The controls borders should "pop up" when the user moves the mouse over the control area and "pop down" after the mouse has left the control area. When the mouse is pressed the border should enter a standard "pressed" state and vice versa when released.

ActiveContainer -

The ActiveContainer control mimics the "docked" forms in the VB5IDE. It is a simple control container (allows the user to place other controls on it) with a "close" box. This control is alignable, so the user may place it on an MDI form. One event has been added: the XBox\_click event. This event occurs when the user clicks on the "close" box. This allows the user flexibility in enabling of disabling the controls iterant closing feature.

ActiveMenu -

The current build of the ActiveMenu control is essentially no different than the ActiveButton Control, except it has a predefined height to be compliant with MS UI standards. For info on the drop-down menu capability, see the MenuButton Control.

ComponentWizard -

The ComponentWizard is a highly sophisticated registry search-algorithm, allowing the user to search the registry and return either some or all of the system's registered OCX's, DLL's, or ActiveX EXE's. To use this control, place it on a form and in which ever form event you wish to trigger this control from type:

vbComponentWizard1.ShowComponents \*Arg1\*, \*Arg2\*, \*Arg3\*

The arguments (arg1, arg2, arg3) are to determine which component types the control should look for. They are each boolean types and arg1 is for OCXs, arg2 is for DLLs, and arg3 is for ActiveX EXEs. This

method initiates the search algorithm and pops up a modal form that displays (alphabetically) the desired components. All the user must do now is select the desired control and click on the "OK" command box. This retrieve's the components Name, Location, and CLSID information and places it in the ComponentName, ComponentLocation, and ComponentCLSID property, respectively. These three properties plus two more (ComponentIndex and ComponentCount) make up the data structure for a collection of selected registered components. ComponentName, ComponentLocation, and ComponentCLSID are control arrays that must be indexed from the ComponentIndex property. Two methods: AddComponent, and RemoveComponent allow the user to manually add or remove Components from the collection. If the user wishes to setup his or her OWN dialog form and routines, the base-level algorithm is available through the ObtainRegisteredComponents method.

#### MenuButton -

The MenuButton control acts as an activeborder drop-down box. The current build allows the user to place a picture on the command box side of the control and write code to handle the click event on the same side. This control has a State property that indicates what state the button is in at any time. The settings for the State property are:

- 0 - No border
- 1 - No border, but the mouse left with the right button still down. (When the menu should be displayed)
- 2 - Active border
- 3 - Active border and the left (main) button is down
- 4 - Active border and the right button is down

This control does NOT contain any inherent forms or menu objects to drop down. These must be implemented as a user-created form, but a template menu form has been created for you and placed in your templates directory. Just refer to the sample project for the source code to implement the drop-down feature.

There is a sample project included to show the use of all of the controls, just unzip the sample.zip file and run it in VB5.

If there are any question about this or other releases, bugs to reports, questions to pose, or comments to submit...then please contact us at: [galwebmaster@hotmail.com](mailto:galwebmaster@hotmail.com) with the words "ACTIVEVB5TOOLS" in the heading.

-Benjamin

Benjamin Crane, CEO  
Galileo Technologies  
[galileotech@juno.com](mailto:galileotech@juno.com)

## HelpScribble

HelpScribble is a help authoring tool written by Jan Goyvaerts. This help file was created with the unregistered version of HelpScribble, which is why you can read this ad. Once the author of this help file is so honest to register the shareware he uses, you will not see this ad again in his help files.

**Recompiling the help project with the registered version is all it takes to get rid of this ad.**

HelpScribble is a stand-alone help authoring tool. It does *not* require an expensive word processor. (Only a help compiler as Microsoft likes keeping the .hlp format secret. Not my fault.)

Here are some of HelpScribble's features:

- The Setup program will *properly* install and uninstall HelpScribble and all of its components, including registry keys.
- Create, edit and navigate through topics right in the main window. No need to mess with heaps of dialog boxes.
- All topics are listed in a grid in the main window so you won't lose track in big help projects. You can even set bookmarks.
- Use the built-in Browse Sequence Editor to easily create browse sequences.
- Use the built-in Window Editor to change the look of your help window and create secondary windows.
- Use the built-in Contents Editor to create Windows 95-style contents files. Works *a lot* better than Microsoft's HCW.
- No need to mess with Microsoft's SHED: use the built-in SHG Editor to create hotspot bitmaps. Draw your hotspots on the bitmap and pick the topic to link to from the list.
- With the built-in Macro Editor you can easily compose WinHelp macros whenever needed. It will tell you what the correct parameters are and provide information on them.
- If you have a problem, just consult the online help. The help file was completely created with HelpScribble, of course.
- HelpScribble is shareware. However, the unregistered version is *not* crippled in any way. It will only add a small note to your help topics to encourage you to be honest and to register the shareware you use.

These options are very interesting for Delphi and C++Builder developers:

- If you are a component writer, use the Delphi Parser to build an outline help file for your component. Just fill in the spaces and you are done. HelpScribble can also extract the comments from your source file and use them as the default descriptions.
- If you are an application writer, HelpScribble provides you with a property editor for the HelpContext property. You can select the topic you need from a list of topic titles or simply instruct to create a new topic. No need to remember obscure numbers.
- The property editor also provides a tree view of all the components on your form and their HelpContext properties. This works very intuitively. (Much nicer than those help tools that simply mess with your .dfm files.)
- HelpScribble can perform syntax highlighting on any Delphi source code in your help file.

HelpScribble is shareware, so feel free to grab your copy today from my web site at <http://www.tornado.be/~johnfg/>

