

Visual Basic Programming with ClassAction™ Lite

Welcome to ClassAction Lite by the Crescent Division of Progress Software (Crescent). ClassAction Lite is a series of Windows 32-bit API (Win32 API) objects for Visual Basic. ClassAction unleashes the power of the Windows API by providing Visual Basic developers an object interface to the API. Instead of declarations and function calls, ClassAction is objects with properties and methods.

ClassAction Lite is a limited edition of Crescent's ClassAction. ClassAction can give you more access to the WinAPI in either 16- or 32-bit environments. At the end of this document is a registration form. If you register this edition of ClassAction, you can upgrade to the full copy of ClassAction for \$69.00 (US). Mail or Fax your registration today!

Basically, if you know how to work with custom controls, you can work with ClassAction's objects. This document provides a brief overview of ClassAction and a guide to its objects, and contains the following sections:

- A brief overview of ClassAction.
- An introduction to the Foundation Object Standard.
- Hardware and software requirements for ClassAction.
- How to get Technical Support for ClassAction.

Let's begin.

Installing ClassAction

Installing ClassAction Lite is a snap, just run the setup program. It will copy the cfo32l.dll file into your windows system directory and the help file into your windows directory. Also, take a look at the help file section [Using ClassAction](#). It describes how to add ClassAction in a VB project.

What is ClassAction?

ClassAction brings order to the Windows API. Without ClassAction, the Visual Basic developer has to wade through complex declarations and function calls. And, in some circumstances, leaf through mounds of documentation to find the magic API call, only to find the syntax written for C++. ClassAction makes sense of the complexities of the Windows API.

In ClassAction, there are objects specifically built for certain tasks. For all the window API functions, there is an object called foWindow where fo refers to the Foundation Object standard. For system functions, there is foSystem. By grouping API functions into logical objects, ClassAction effectively gives you an API library. Best of all, you already know how to use ClassAction.

ClassAction's Objects

ClassAction groups API functions into objects. The following is a list of objects that are available ClassAction.

Object Name and Description

foBitmap - The foBitmap object lets you attach, detach, load or delete a bitmap. You can also use properties to update or interrogate the bitmap.

foBrush - The foBrush object lets you attach (based on its color), detach, or delete a Brush. You can also use properties to update or interrogate the status/type of Brush attached using hBrush property.

foButton - The foButton object lets you update or interrogate the button with the Value property.

foColors - The foColor object lets you use properties to update or interrogate the color of various user interface objects including: buttons, borders, captions, menus, and scrollbars.

foComboBox - The foComboBox object lets you perform TextBox-specific actions like locating a specified text string. In addition, you can also use properties to update or interrogate the Combobox.

foDeviceContext - The foDeviceContext object lets you draw shapes, select fonts and bitmaps, and attach windows, etc. In addition, you can also use properties to update or interrogate current values for the brush, font, icon, etc.

foDrive - The foDrive object lets you use properties to update or interrogate information about sectors and clusters.

foFont - The foFont object lets you assign fonts by attaching to default, OEM, or ANSI fonts. You can also detach from or delete fonts. In addition, you can use properties to update or interrogate the current font.

folcon - The folcon object lets you attach, detach, load and delete icons.

foListBox - The foListBox object lets you perform ListBox specific actions like locating a specified text string. In addition, you can use properties to interrogate or update an item's data, or height.

foMemory - The foMemory object lets you manipulate memory. For example, you can allocate memory and copy a block of memory to or from a specified destination, etc..

foMetrics - The foMetrics object lets you update or interrogate various user interface objects including height and width (properties) of user interface objects like: borders, frames, scrollbars, menus, screens, etc.

foPalette - The foPalette object lets you attach, detach and delete palettes. In addition, you can use properties to interrogate or update the Palette.

foPen - The foPen object lets you create, delete, attach, and detach pens. In addition, you can use properties to update or interrogate the pen's color, style, and width.

foPoint - The foPoint object lets you set the offset and the window for a point. In addition, you can use properties to interrogate or update the point's x and y location.

foRectangle - The foRectangle object lets you specify the size, location and fill for a rectangle object. In addition, you can use properties to update or interrogate the exact location of the sides of the rectangle.

foRegion - The foRegion object lets you create, delete, attach and detach a region. In addition, you can use properties to update or interrogate the region object through the hRegion property.

foSize - The foSize object lets you update or interrogate size through the x and y objects.

foSystem - The foSystem object lets you specify certain system functions like colors, MessageBoxes, Metrics, etc. You can also use properties to update or interrogate the system settings like the active window, the desktop window, the computer name, etc.

foTextBox - The foTextBox object lets you perform TextBox-specific actions like undo, line count, and replace selection. In addition, you can also use properties to update or interrogate whether

undo is supported, whether the TextBox is read-only, and whether the TextBox was modified.

foWindow - The foWindow object lets you perform Window-specific actions like setting focus, setting Winhelp, enabling scrolling, etc. In addition, you can also use properties to update or interrogate the window's size, it's class name, it's children, it's text, etc.

ClassAction's Properties and Methods

ClassAction uses properties and methods. Just like custom controls (VBXs and OCXs), ClassAction's objects can be controlled by accessing properties and calling methods. For example:

If you use Visual Basic's List Box control, you have probably added items to the list by:

```
List1.AddItem "Apples"
```

Where List1 is the control name, AddItem is the method that you are calling, and "Apples" is the item that is being added to the list. Like the List Box (as well as all Visual Basic's objects and controls), ClassAction's methods can be called in the same manner:

```
foWindow.DragAcceptFiles True
```

foWindow is the object, DragAcceptFiles is the method, and True is the parameter that is being passed to the method.

ClassAction's objects also have properties. ClassAction object properties are read and assigned like a custom control. If you read the property of a control, you get it's value. If you change the property of a control (provided it's not read-only), the control will change. For example:

The Height property of a control will change the operational height of the control:

```
List1.Height = 500
```

List1 is the control, Height is the property, and 500 is the value being assigned. With ClassAction, properties are assigned similarly:

```
foWindow.Height = 500
```

foWindow is the object, Height is the property, and 500 is the value being assigned. Once you have installed ClassAction, you must create a reference for the ClassAction DLL. Once the reference is made, you can create any instance of a ClassAction object in your project by declaring a new variable using the object you want as your type. For example:

```
Dim MyWindow as New foWindow
```

Will create an instance of the foWindow object. In your code, you can now access the MyWindow object's properties and methods:

```
MyWindow.Height = 500  
MyWindow.Flash True  
MyWindow.BringToTop
```

And so on...

For every ClassAction object you wish to use, you must create a variable for that object. Like other variables in Visual Basic, your ClassAction object variables can be Private, Public, Local or

Global. You decide the scope.

As you can see, ClassAction operates like any custom control or object in Visual Basic. We've unlocked the API for you. Now it's your turn to take full control of your Windows development.

Hardware and Software Requirements

ClassAction has the following hardware and software requirements:

- Any IBM-compatible machine with an 80386 processor or higher
- A 3 1/2" floppy drive or CD-ROM Drive
- Microsoft Windows® 95, Windows NT™ Workstation version 3.51 or later, or Windows version 3.1 (requires MS-DOS® version 5.0 or later) operating system
- 8 MB of memory (16 MB or more recommended) if using Windows 95 or Windows 3.1; 16 MB if using Windows NT Workstation
- Visual Basic 4.0 Standard, Professional or Enterprise Edition
- 5 MB of hard-disk space

ClassAction Technical Support

The Crescent technical support staff is ready to help you with problems that you encounter when installing or using ClassAction.

If you need technical support, contact Crescent using any of the following methods:

- By Telephone - Contact Crescent's North American technical support staff at: (617) 280-3000 -- Monday through Friday from 9:00 a.m. to 5:00 p.m. EST.
- By FAX - Contact Crescent by FAX at: (617) 280-4025.
- Via BBS - Contact Crescent through our 24-hour bulletin board service at: (617) 280-4221.
- Via CompuServe - Contact Crescent through CompuServe address: 70662,2605 Crescent also maintains a section in the MS Windows Components A+ Forum on CompuServe. To reach the Crescent section, type the following at the CompuServe prompt: GO CRESCENT
- By Electronic Mail - Contact Crescent using the Internet: crescent-support@progress.com
- Via the WWW - View the Crescent Web page at: <http://www.progress.com/crescent>
- By Mail - Address your correspondence to: Technical Support, Crescent Division, Progress Software Corporation 14 Oak Park Bedford, Massachusetts 01730

Please have your product name, version number, serial number, and system configuration information available so that the Crescent technical support staff can process your support requests as efficiently as possible.

Copyright © 1995, 1996 Progress Software Corporation
Copyright © 1995, 1996 ViewPoint Technologies, Incorporated

Crescent ClassAction™, and ClassAction™ Lite are trademarks of Progress Software Corporation

All company and product names are the trademarks or registered trademarks of their respective companies.

ClassAction Lite Registration

First Name:

Last Name:

Title:

Company:

Street or PO Box:

Dept./Suite:

City:

State/Province:

Zip/Postal Code:

Country:

Telephone:

Fax:

E-Mail:

Mail or Fax your registration to:

Crescent Division of Progress Software
ClassAction Lite Registration
14 Oak Park
Bedford, MA 01730

Fax: 617-280-4025