

Brandable 32-bit Applications

The Future of Software Branding

by Roy D. Pope, Jr.
Blue Aegis Software & Marketing

<http://www.BlueAegis.com>

Copyright © 2005 by Blue Aegis Software & Marketing
All Rights Reserved

This ebook has been developed and released for free distribution. You may make as many copies of this ebook as you desire and distribute those copies without charge. This ebook may be bundled with other software and ebook products for which a monetary charge is made.

Introduction

A few years ago, I purchased an ebook titled 'The Road Map to Internet Success' by Donny Lowy. I was excited about this fine ebook for a number of reasons, including the fact that I could "brand" my web site information into the ebook and resell it for royalty-free profits!

In the years to follow, I spent several hundred dollars purchasing several thousand ebooks with resale rights. Like Donny's ebook, most of these were also brandable with all manner of information. I could brand them with anything from my web site details to my affiliate identification codes for products and services that I was also reselling.

As time progressed onward, I began to acquire a new form of ebook – the ebook composed of javascript generators and marketed as brandable software products. And then, there were a couple of ebooks containing javascript for games such as solitaire and hangman – and they were marketed as brandable games.

I closed my wallet and opened my mind to new thoughts.

People are actually publishing ebooks filled with javascript to simulate software products because ebook compilers allowed them to make their end-products brandable!

I have met some of the authors of javascript based "pseudo-applications" through online bulletin boards and chat rooms. Most seem to be relatively intelligent, and a few are indeed programmers adept in developing applications in such languages as Delphi and Visual Basic.

When I asked them why they developed brandable "pseudo-applications" using HTML, javascript and ebook compilers instead of developing true 32-bit applications with branding features using a higher language, they all agreed on one thing: there's no easy way to brand a 32-bit application.

32-bit Branding Made Easy

In late January 2005, Blue Aegis Software & Marketing released onto the market the Brandable Application SDK for Visual Basic 6.0. To my knowledge, this is the first Software Development Kit to ever be developed which would enable programmers in any language to create fully brandable 32-bit applications.

No longer are programmers confined to developing pseudo-applications driven by javascript in order to take advantage of the widely popular branding craze sweeping the internet. No longer are they restricted to the development of simple javascript generators and javascript board games.

With the Brandable Application SDK for Visual Basic 6.0, those programmers adept in software development using Visual Basic are now able to give all of the software that they develop branding features!

By simply referencing a small DLL (Dynamic Link Library) included with the SDK in a Visual Basic project, then adding a single line of code to the project, the application immediately inherits brandable features when compiled into an executable. It's then a simple matter of adding special "placeholders" within the project in order to make use of the brandable features.

The SDK includes a special utility aptly called the Master Application Branding Utility (MABU), and a template set called the Application Branding Utility Setup Template (ABUST). Once the brandable application has been built, compiled, and packaged using the Package and Deployment Wizard, the developer runs the MABU which uses the ABUST to create a unique, customized Application Branding Utility (ABU) for the brandable application.

Along with the ABU created by the MABU is an archive file called the Brandable Application Setup Template (BAST). This file contains the "unbranded" version of the brandable application, along with all files necessary for the building of the application's setup package. The ABU and the BAST are the files that are distributed as the "brandable" application.

When the ABU is run by the end-user, it extracts the files from the BAST and performs a long series of operations on those files. Amongst those operations is the task of writing the information that the end-user provides in the ABU to the brandable executable file. Once the executable file has been modified to include the end-user's information, all files extracted from the BAST are then assembled together into a neat little setup package ready to be distributed as a "branded" application.

What can be branded to an application?

The Brandable Application SDK for Visual Basic 6.0 was originally developed to give software developers a means of building a strong reseller base for their software products. When used as originally intended, a developer's application reseller will be able to brand his company's name, web site URL (Uniform Resource Locator), street address, telephone number, or other contact information into the application. This will identify the reseller as a legitimate reseller of the product and place in front of his customer his contact information every time the branded application is run on the computer.

Of course, simple reseller identification information is the least of things that can be branded to an application using the SDK.

Authors of brandable ebooks have been using branding to amass huge fortunes through techniques that they have coined as "viral marketing". Now, the ability to create

brandable 32-bit applications places the awesome power of viral marketing into the reach of programmers who would rather develop real applications instead of pseudo-applications.

With a little bit of programming effort, brandable applications can be developed which either serve clickable banners and advertisements from an internal database or call them from an active internet connection. The developer can easily configure the application so that, when branded, the clickable banner gives credit to the brander as an affiliate of the advertised site, product or service. And this credit could benefit the developer as well as the person branding the application if the affiliate program pays on multiple tiers.

Another neat little way to take advantage of application branding is to develop an application so that in an unbranded state certain features of the application are disabled or hidden. Once properly branded, those disabled or hidden features become enabled or visible.

This would permit the developer to build one application and distribute it as a demonstrative (demo) copy and, at the same time, as a full version copy that only needs to be branded to be converted from its demo state.

In fact, there are limitless methods of using application branding to the benefit of the developer, the reseller, the end-user, and everyone in-between. Constant values can be branded to an application to change the outcome of computations or to customize generated reports; encryption keys can be “burned” into executables for heightened security; email addresses and web site addresses can be secretly embedded into internet applications to intercept information from the end-user’s computer. The list never ends.

Is it really branding the executable?

I was in communication with a Visual Basic developer for a while who wrote to me that he had once developed branding technology for his applications. To brand his applications, the contents of an initialization (INI) file were modified by the branding utility. These contents were then read by the “brandable” application when it was run and loaded into the application’s variables.

I could only shake my head. This was not application branding! This was merely text-file writing and reading.

The Brandable Application SDK for Visual Basic 6.0 was developed to actually write the branding information directly inside of the executable application. This information, when written, is encrypted. The encryption is to deter wannabe hackers with hex editors from changing the branded information to reflect their own information in an attempt to prevent piracy of branded applications.

Yes, 32-bit executable branding is a reality. Yes, the end products that developers can now produce with the aid of the SDK are now genuinely brandable.

The future of brandable applications

The ability to create brandable 32-bit applications has opened a fantastically new door on the future of software and computing. As I write this small ebook, I am already at work developing a much more advanced version of the Brandable Application SDK brimming over with features that were once only available to actors playing parts in sci-fi movies.

The ability to create brandable 32-bit applications means **power** – and with power comes the need for responsibility.

Imagine, if you will, the ability to create a software product that appears and behaves as one product, but which transforms itself into another more powerful software product once a specific date in time has passed or the product has shelled a certain executable component. A simple site indexing robot could suddenly morph into a file harvester, or an accounting application could find itself transformed into an agent of corporate espionage.

Brandable application developers could create internet applications which would, of course, be given access to the internet through firewalls. These internet applications, when branded, could then replace adware components and spybots, giving a whole new angle to be played by data mining facilities and advertising networks.