



Visual Basic Script and the Access Developer

Ken Lassesen

Visual Basic Script (VBS) is a subset of Visual Basic for Applications (VBA) designed for use in Hyper Text Markup Language (HTML) files on the Internet and intranets. The commands available in VBS are generally identical to the commands in VBA. Ken explains some fundamentals of the philosophy behind the design of VBS, how its arrival may impact Access developers, and provides examples of using DAO and Access in HTML files. VBS is installed with Microsoft Internet Explorer 3.0.

The key element of Visual Basic Script is safety from the unknown. VBS is designed to allow safe programming on the Internet. Allowing an unknown person to load and execute a program on your computer can easily become a tragic mistake. To prevent random acts of terror, strong safeguards are needed with any Web-enabled language. Web-enabled languages can approach safety in several ways. The two dominant methods-the sandbox approach and the accountability approach-have the same goals but different consequences.

The sandbox approach

Sun Microsystems Inc. took the sandbox approach in the development of Java. They made the language incapable of harming the PC by building a limited operating system (sandbox) which restricted the language. To guarantee complete isolation from the real hardware, the compiled code from the application must not use any real hardware commands; effectively, the app runs on a completely virtual machine.

This approach can be very advantageous. Your application can run on any type of computer, from your old Atari game machine to a Cray supercomputer. (Historically, Java was created as a programming language for toasters-honest!) All someone has to do is port the Java virtual operating system to the target computer and your application will run. Like anything, however, the sandbox approach carries with it a few disadvantages:

Speed: Your application runs a lot slower than with native or p-code.

Lameness: You can't do any *real* work (you only have sand to work with).

Although it's possible for Java to touch the operating system, this action violates the intent of the "sandbox" and requires some form of accountability. For example, Microsoft's Visual J++ has samples using DAO.

The accountability approach

The accountability approach doesn't attempt to hide to avoid doing real work; instead, it builds in legal accountability. Software that can touch your hardware must clearly identify its origins and its degree of risk to your PC. The legal identity of the software source must be established and the file verified to be unaltered. This is the model used by Microsoft in developing Visual Basic for Scripting with its Internet Explorer.

The first component of the accountability approach is VBS, a subset of VBA with a long list of commands *removed* and the number of data types *reduced* to produce a small, compact, safe language. Some activities that are denied-to protect your PC-are:

File access.

Access to the clipboard.

Access to Dynamic Data Exchange (DDE).

Ability to create or get objects.

Ability to use the SendKeys command.

In addition, to reduce the size of the runtime, various non-essential elements were removed:

Error handling, except for On Error Resume Next.

Collections, non-variant data types, and financial functions.

Complex control-flow commands.

The result is a small, compact subset of VBA that could easily be ported to other operating systems (Macintosh and UNIX). The difference in size is apparent when you look at the file sizes: the VBA32.DLL and VBA232.DLL files are 736K and 1,025K respectively, while the VBScript.DLL and Jscript.DLL files take up only 202K and 242K respectively.

The second component of this approach is support for OCXs (recently renamed as ActiveX controls). By adding OCXs, any type of activity can be done on the PC. The mature and popular OCX technology allows a fast and familiar jump start to VBS. All of the features that were removed from VBS can be added back through ActiveX controls. Wait a minute! If all of the features are added back in, how can this be safe?

The script is safe because the developers of the controls have identified themselves as a legal entity, prevented anyone else from tampering with the control, and made claims on the safety level of the control. For an ActiveX control to run smoothly in a HTML page, the ActiveX controls must be:

digitally signed.

marked Safe-for-Scripting.

marked Safe-for-Initialization.

If the controls are missing any of these features, the control won't initialize with the default settings of Internet Explorer. Let me explain these three magic safety bullets.

Digitally signed means the control contains a special certificate that's issued by a third party. This certificate attests to the legal identity of the developer and confirms that the control has not been altered since it was signed. The legal identity of the control publisher has been verified by an independent organization. The publisher has accepted responsibility of being legally identified with any software signed by the certificate. If the software does nasty things to your PC, the publisher is potentially liable.

Safe-for-Scripting means that the control can't be made, through scripting, to do anything harmful to your PC, nor can it be used to extract any information. This isn't an "under normal use" assertion by the developer-it's a claim that the control is safe under any circumstances through VBS. If it isn't, there may be a breach of contract and damages can be sought against the developer.

Safe-for-Initialization means the control can't be made, through the initial values assigned to its

properties, to do anything harmful to your PC, nor can it be used to extract any information. This is the same type of assertion as described for Safe-for-Scripting.

Database and Visual Basic Scripting

Data and information processing are what Access is about. If VBS prevents read, write, or update abilities to databases, then why write about it? Part of the accountability model is the ability to obtain a certificate from an HTTP server. This server certificate states that, although you may be using ActiveX controls that aren't digitally signed or marked for safety, you promise not to do anything inappropriate. Typically the HTTP server is an intranet server containing pages produced by you.

You can access data through DAO and through OLE Automation of Access from an HTML page running VBS. The advantage of doing it through an HTML page is that you can automatically upgrade components and applications without having to uninstall and reinstall applications on each PC. If you modify your application, the modified version will be available everywhere when you place it into production. Not only will it be available-it'll be the only version of the application running!

How does this happen? The HTML page may contain a tag called Object that identifies an ActiveX Server or Control. Part of this tag is named CodeBase and points to a Diamond Cabinet file (CAB) and gives a version number. If Internet Explorer finds that the version installed on the PC is older than the version specified by CodeBase, it will download and install the files in the CAB. No setup program. No uninstalling. If Internet Explorer doesn't find the ActiveX Server in the registry, it will then install the CAB automatically.

Practical examples

Time and paper prevent me from teaching you everything in this article. If you want more information, check out IDG's new book, *Introducing VBScript and ActiveX* (warning: my wife co-authored it with Ken Spencer and Ken Miller so I may be biased). I'll assume you have some familiarity with HTML and ActiveX controls.

The following simple HTML page allows me to access any part of the DAO Automation Server from an HTML file. The HTML Object tag is effectively an enhanced CreateObject call that uses the ClassID instead of a mnemonic name like "DAO.DBEngine." I used my article on the MSDN Library, "Using Microsoft OLE Automation Servers to Develop Solutions," to look up the DAO.DBEngine object and then used the ClassID listed there:

```
<HTML><BODY>
<OBJECT ID="DAO" CLASSID=
  "CLSID:00025e15-0000-0000-c000-000000000046">
</OBJECT></BODY></HTML>
```

Although VBS doesn't support GetObject or CreateObject commands, you create most objects by putting the object's ClassID in the Object Tag. If you use the ActiveX Control Pad to write HTML, when you click the Script Wizard you can see the hierarchy of DAO in front of your eyes! Figure 1 shows the Script Wizard in action.

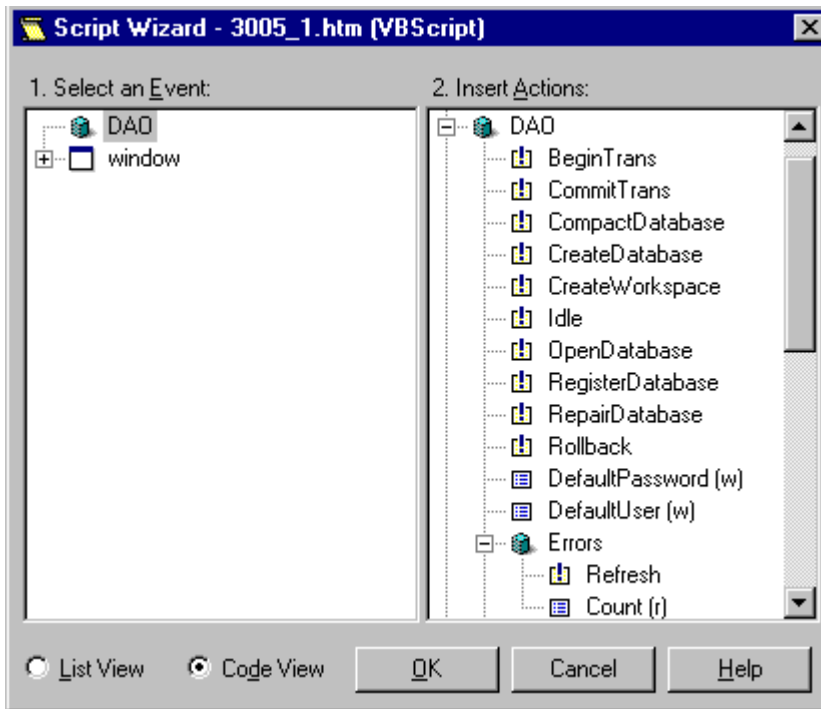


Figure 1. The DAO object in the ActiveX Control Pad Script Wizard.

For example, if you need to create a new Access database simply add some VBS to the page. The VBS must be enclosed in Script tags so Internet Explorer knows whether to display or execute the contents. The first comment line shows what's inserted when you click CreateDatabase in the Script Wizard shown in Figure 1. It's similar to the code written by the Access Object Browser except you can't use named arguments. Similarly, named constants aren't available. Looking up the values in Access' Object Browser quickly produced the following code to add to the HTML page:

```
<SCRIPT LANGUAGE="VBScript"><!--
Sub Window_OnLoad()
'Call DAO.CreateDatabase(Name, Connect, Option)
Call DAO.CreateDatabase("NEWDB.MDB", ";LANGID=0x0409;
  CP=1252;COUNTRY=0", 2)
End Sub
--></SCRIPT>
```

The HTML page is called vbs1.htm. Load it into Internet Explorer and then . . . we need some more explanations.

DAO isn't marked as Safe-for-Scripting so the VBS code won't execute with Internet Explorer's default safety setting of High. Changing the safety level to Medium permits the VBS code to execute, but the dialog box shown in Figure 2 appears first.

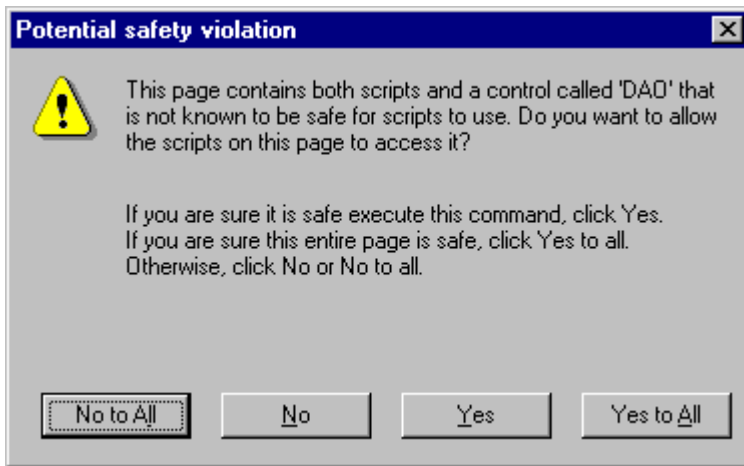


Figure 2. Internet Explorer's "potential safety violation" dialog box.

If this HTML file came across the Internet from an unknown source, you'd click No to All automatically. You'd never click Yes because the script *could* do anything that an Access application could do in this case. If you click Yes for my sample page, the HTML page creates a new Access database. (You may need to perform a File Find to locate it on your PC.) Of course, you could continue creating tables, extracting data, and so on, but that's normal Access development at this point. Once you have an instance of the DBEngine object, you can drill down to any child object.

The next example shows you how to access the Access object from an HTML page. The Access object is available with Access 95 and higher; Access 2.0 and lower can't be controlled by OLE Automation. Returning to my MSDN writings to get the ClassID, you quickly create a new page (vbs2.htm). You could've used the registry to locate the class ID but that usually means some trial and error to correctly identify things. All that you needed to do was change the ClassID (and the ID was a courtesy), and you're ready to use Access:

```
<HTML><BODY>
<OBJECT ID="MSAccess"
CLASSID="CLSID:B54DCF20-5F9C-101B-AF4E-00AA003F0F07">
</OBJECT></BODY></HTML>
```

When you load this HTML page into Internet Explorer, you'll see Access load itself as the HTML page creates an instance of Access. The user can't prevent Access from appearing when you load this page. Even with Internet Explorer's safety level set at High, Access will load. The safety levels prevent the properties of ActiveX controls from being set and the object (ActiveX Server) from being scripted (a.k.a. OLE Automation). Access loads but can't be used when the safety level is set at High.

Unlike DAO, the Script Wizard doesn't show any methods or properties. What the Script Wizard will show depends on how the ActiveX/OLE Automation Server was created. To illustrate that you can control Access, I added the following simple and very safe script:

```
<SCRIPT LANGUAGE="VBScript"><!--
Sub Window_OnLoad()
  MsgBox MSAccess.Eval("5*8-7")
End Sub
--></SCRIPT>
```

After you click Yes to the safety warnings, the message box shown in Figure 3 appears.

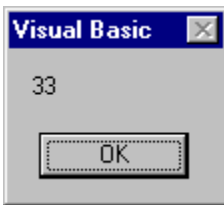


Figure 3. The message box produced by a simple script.

Now I should point out the danger of setting the safety level to None. If you do and then you happen to hit an evil page on the Internet that uses Access, you could permit someone to:

Connect to any application using DDE.

Use the Clipboard.

Transfer data.

Write VBA code and then execute it.

Read your .INI files and registry.

Read your e-mail.

Send threatening e-mail to every member of the U.S. Congress with your name on it.

In short, you may be in big doo-doo.

Summary

In this article I've introduced you to Visual Basic Script on the client side. VBS is also available on the server side with Microsoft's Internet Information Server and may be licensed from Microsoft for use with your own products (see <http://internet/vbscript/us/vbsmain/hosting.htm>). VBS with Internet Explorer will become a powerful ally in developing corporate applications because of the automatic installation and update capabilities it provides. Access databases can be connected to ActiveX controls, providing opportunities to better use data and greater user flexibility. These features come with a significant safety risk to users unless the philosophy and nature of the Internet are understood.

DOWNLOAD LASS01.EXE at www.pinpub.com/access

*Ken Lassesen is well-known for his articles on the MSDN Library and his talks at Tech*Ed. He's been programming since 1968 and working with relational databases since 1980. Ken is currently working as a software design engineer for the Microsoft Network. He is married to Lauren Lassesen, a Web design engineer and editor of the webzine ActiveX Journal for HTML Writers at <http://www.folkarts.com/journals/activex/>.*

To find out more about Smart Access and Pinnacle Publishing, visit their website at <http://www.pinpub.com/>

Note: This is not a Microsoft Corporation website. Microsoft is not responsible for its content.

This article is reproduced from the January 1997 issue of Smart Access. Copyright 1997, by Pinnacle Publishing, Inc., unless otherwise noted. All rights are reserved. Smart Access is an independently produced publication of Pinnacle Publishing, Inc. No part of this article may be used or reproduced in any fashion (except in brief quotations used in critical articles and reviews) without prior consent of Pinnacle Publishing, Inc. To contact Pinnacle Publishing, Inc., please call 1-800-788-1900.

[Send feedback](#) to MSDN. [Look here](#) for MSDN Online resources.