INTERACTIVE DEVELOPER

OLE Makes Its Way to the Internet

Glimpse Microsoft's document-centric OLE vision and how "Internet OLE" fits into VB's future.

by Roger Jennings

Microsoft's Internet strategy, outlined last December 7 by Bill Gates, Paul Maritz, and Pete Higgins, nearly caused a meltdown of Microsoft's World Wide Web site. Computer journalists, security analysts, Visual Basic programmers, and just about everyone else with a Web browser converged on http://www.microsoft.com to download the technical details behind the public relations hype that's become de rigueur for big-time product announcements (see Matt Carter's Interactive Developer column in the February 1996 issue of *VBPJ* for more information about the role of Visual Basic in Microsoft's Internet initiatives). Initial dispatches from the computer press concentrated on Microsoft's new cross-cultural ties, such as the letter of intent to license Sun Microsystem's Java programming language and runtime JavaScript, plus the agreement to cross-license Oracle's PowerBrowser OLE control and Microsoft's compact version of VBA, Visual Basic Script (VBS). Bill Gates' willingness to make deals with arch-enemies Scott McNealy of Sun and Larry Ellison of Oracle glues a veneer of openness to Microsoft's Windows-centric view of the Internet. Behind the scenes, however, Microsoft's gearing up to make OLE 2.x the backbone of its Internet and intranet strategies. Today, 70 to 80 percent of all Web browsers run under Windows 3.x and Windows 95. Microsoft clearly intends to make OLE the linchpin of their efforts to protect Windows' market position against impending competition from the likes of Oracle's $500 "Network Computer" and similarly priced diskless consumer PCs based on Apple's new Pippin architecture. This month's column gives you the big picture of Microsoft's document-centric OLE vision and how "Internet OLE" fits into the future of Visual Basic and its Internet derivative, Visual Basic Script.

Roger Jennings, a principal of OakLeaf Systems, is a consultant specializing in Windows client/server database front ends. He has more than 25 years of computer-related experience, and is the author of four programming books: Using Access 95, Special Edition, *and* Discover Windows 3.1 Multimedia *for Que Books; and* Access 95 Developer's Guide *and* Database Developer's Guide with Visual Basic 4 *for Sams Publishing. He's also the coauthor with Peter Hipson of Sams'* Database Developer's Guide with Visual C++ 4. *Roger's now in the process of writing* Using Windows NT Server 3.6, Special Edition, *and* Using Windows Desktop Video, Special Edition, *for Que. Reach him by CompuServe at 70233,2161 or on The Microsoft Network as Roger_Jennings.*

MICROSOFT PLAYS A FAST GAME OF CATCH-UP

Microsoft may have missed the first boat to the Internet, but that doesn't mean that Bill Gates and Co. aren't up to snuff on the technology side. Despite the accolades heaped on Java by the computer and business press, Microsoft's no stranger to the downloadable objects business. For example, Microsoft Interactive TV (demonstrated at the National Cable TV Association's Cable '95 conference in mid-1995) is based on a distributed version of COM, the Component Object Model that's the foundation for OLE 2.x (see my Interactive Developer column in the October 1995 issue of *VBPJ* for more information on OLE and MITV). MITV servers provide class stores for downloadable executables and DLLs, and object stores for networked OLE components to support interactive TV set-top boxes. The Microsoft/Visa Secure Transaction Technology (STT) for Internet commerce first appeared as a component of MITV. The whitepaper, "Microsoft Interactive Television in Detail," printed for the Cable '95 show, proposed Visual Basic and Visual C++ as alternative programming languages for the applications layer for set-top boxes. Microsoft announced at its Interactive Media Conference in mid-July 1995 an Internet version of its Blackbird authoring tool for The Microsoft Network that incorporates VBA and supports OLE controls. So, much of what Microsoft announced at its December Internet Strategy Workshop was already in the works at Redmond. Microsoft's objective now is to accelerate moving the distribution medium from MITV and MSN to the Internet.

Microsoft's catch-up plan involves these basic browser-side components:

- Win32 Internet client WinInet functions in WININET.DLL.
- Document Objects for creating Internet-enabled OLE servers and containers.
- OLE Scripting with VBS (and Java/JavaScript) for embedded programming in HTML documents.
- OLE controls for customizing HTML documents and adding browser capability to applications.
- Progressive rendering of JPEG graphics.
- ActiveVRML for 3-D animation.
- Internet Studio (the new name given the Internet version of Blackbird), a "visual publishing" tool for developing World Wide Web content.
- A proposed "digital signature" system to verify the integrity of executable applications and objects distributed over the Internet.

The Win32 API for Internet client applications is called Sweeper SDK. When this column went to press, the Sweeper SDK included documentation for WinInet and OLE Document Objects (DocObjects), plus a demonstration application of a sample Framer application that hosts DocObjects. You can read the current lowdown on the Sweeper SDK and download the Framer demo app at http://www.microsoft.com/intdev.

On the server side, Microsoft Internet Information Server (formerly Gibraltar), Merchant secure transaction server,

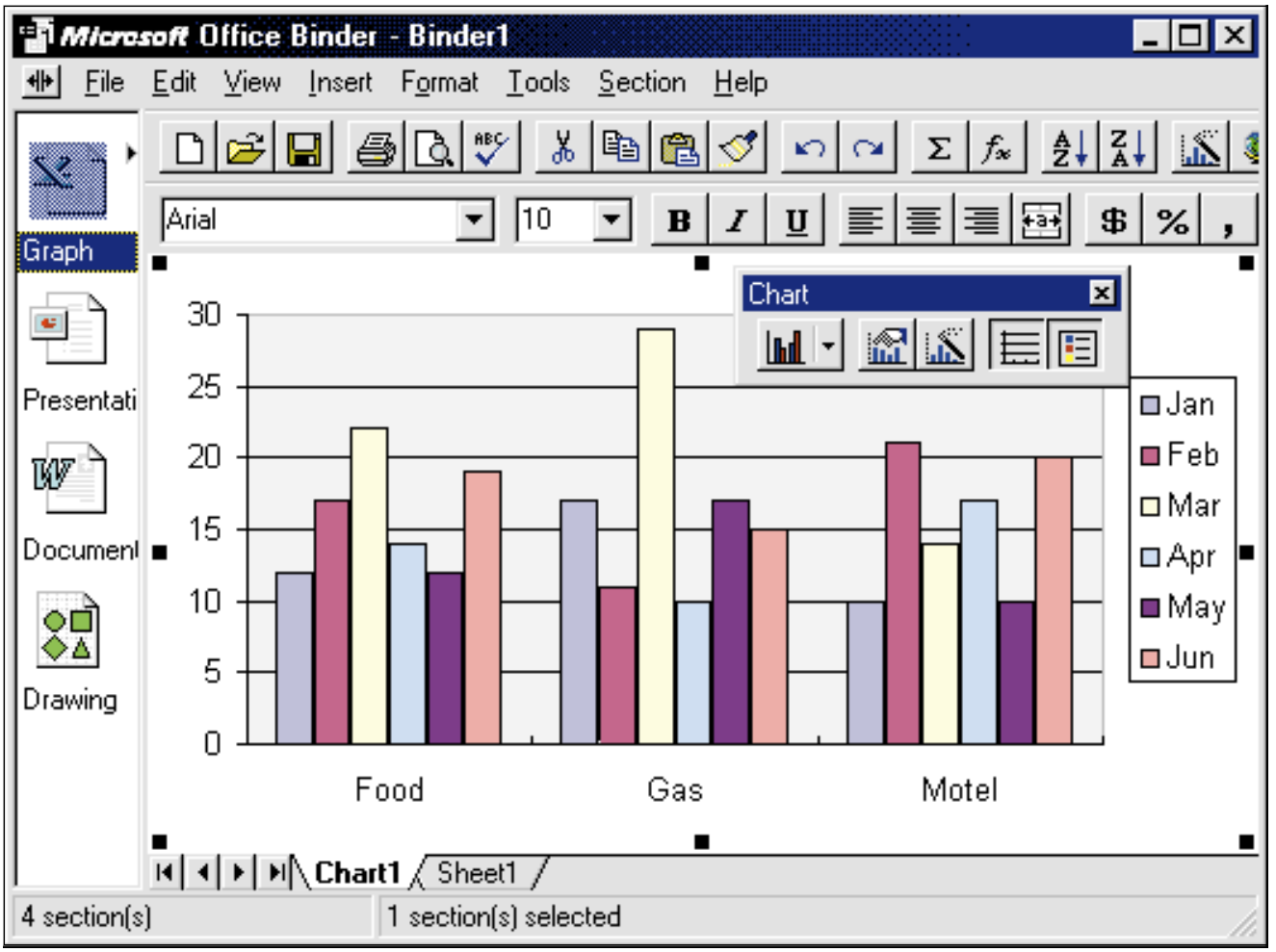

FIGURE 1 ***Documents in a Bind.*** *Microsoft Office 95's Binder is a DocObject container that supports all of the primary features of version 1.0 of the OLE Document Object specification. Web pages that act as DocObject containers can incorporate objects supplied by DocObject-compliant OLE 2.x servers. Substitute a tree view of local and networked folders for the row of icons and you have a future version of Explorer that activates objects in its own DocObject container pane.*

Internet proxy/firewall server (code-named Catapault), and Media Server (code-named Tiger) are scheduled to become components of BackOffice during the first half of this year. Bill Gates announced in his kickoff speech at the Internet Strategy Workshop that Media Server will be adapted to deliver video and audio over the Internet. Microsoft and Process Software Corporation, a Web server ISV, are cooperating on the Internet Server API (ISAPI), which extends the Internet Information Server for commercial applications and system administration. A future version of ISAPI will include ODBC extensions for database connectivity. Version 1.0 of the Internet Information Server should be released as a standard feature of Windows NT Server by or shortly after the time you read this column. Publishing limitations prevent full coverage of all of these forthcoming products, so I'll concentrate on the components that primarily are designed to support and extend OLE 2.x in the Internet and intranet environments.

STANDARDIZE INTERNET ACCESS WITH WININET

WININET.DLL (WinInet) is a 32-bit add-on Sweeper library that provides operating system support for client-side Internet functionality. It's a good bet that the future Internet-enabled versions of Microsoft Office 95 will make extensive use of the WinInet functions. Initially, Microsoft is making WinInet available to developers as a distributable add-in for Windows 95 and Windows NT 3.51+; WinInet will be incorporated in future versions of Windows 95 and Windows NT. The primary objective of the library is to eliminate the need to deal directly with TCP/IP, Internet protocols (such as FTP and HTTP), and Windows Sockets. WinInet includes ANSI and Unicode versions of Internet..., Ftp..., Gopher..., and Http... functions that support reentrancy for creating multithreaded applications. Many of the WinInet functions will replace a substantial amount of C programming now required to implement routine tasks, such as scheduled retrieval of files from an FTP server or automatically gathering daily information from a specific set of Web pages. Application programmers can use WinInet functions to integrate Internet functionality within Windows productivity applications. You can download the preliminary specification for WinInet as http://www.microsoft.com/intdev/inttech/docobj.htm.

The basic WinInet functions include:

- InternetOpen, which returns a HINTERNET handle.
- InternetConnect, which opens a connection to a specified server.

• InternetOpenUrl, an alternative to InternetConnect; it begins reading a specified HTTP, FTP, or Gopher site.
• InternetReadFile, which fills a buffer with content.
• InternetWriteFile, which writes data from an FTP server to a buffer.
• InternetGetLastResponseInfo, which handles errors.
• InternetSetStatusCallback, which provides information for display in a status bar.
• InternetCloseHandle, which closes an HINTERNET handle opened by InternetOpen, InternetConnect, or InternetOpenUrl.

HttpOpenRequest and HttpSendRequest deal with specific HTTP headers.

Visual Basic developers can declare WinInet function prototypes and use all but the callback functions in conventional VBA code, but it's likely that OLE controls with hooks to appropriate WinInet Internet..., Http..., Ftp..., and Gopher... functions will be available to simplify Visual Basic 4.0 and Access 95 Internet programming. An alternative is to write a simple in-process or out-of-process OLE server with Visual Basic 4.0 or, preferably, Visual C++ 4.0 to provide an OLE wrapper around WinInet for retrieving Web pages or files. It shouldn't surprise anyone that all of Microsoft's Internet-related additions are 32-bit-only. Microsoft is giving away the bulk of its Internet technology, so recouping the Internet investment depends on increased sales of Windows 95, Office 95, and Windows NT.

CREATE COMPOUND BROWSERS WITH DOCOBJECTS

OLE DocObjects began life as the proprietary technology behind the Microsoft Binder included in Office 95. All members of Microsoft Office 95 (except Schedule+) and Visio Corp.'s Visio 4.0 are DocObject servers. Binder is an enhanced OLE 2.1 DocObject container that can handle multiple objects, called Sections. Unlike Visual Basic's OLE 2.0 OCX, Binder supports multiple views of an object within a Section (see Figure 1). For example, in Binder you can view embedded documents in Word 95's Normal, Outline, Page Layout, and Master Document mode; Visual Basic's OLE 2.0 OCX limits you to Word's Page Layout view unless you open Word. The reason for this limitation is that the frame of the OLE 2.0 OCX is foreign to the server. That is, Word, Excel, PowerPoint, and other OLE 2.x servers don't "own" the frame in which to display an embedded document. Lack of ownership contributes to difficulties displaying the server's frame adornments, such as scroll bars, workbook tabs, and toolbars within the container. These problems limit the utility of VB4's OLE 2.0 OCX with complex embedded objects, such as Excel workbooks. DocObjects abstract views, so that the server owns the container-supplied view frame (for adornments) and the document content view within the container. Compare the behavior of a Word or Excel Section of a binder with the same Word or Excel object embedded in the OLE 2.0 OCX or inserted into a Visual Basic form to see the viewing improvements offered by activated DocObjects. It wouldn't be too surprising to see a 32-bit DOCOBJ32.OCX (or whatever) control for Visual Basic 4.0 and Access 95 developers "real soon now," plus forms that fully support insertable DocObjects in a future version of Visual Basic.

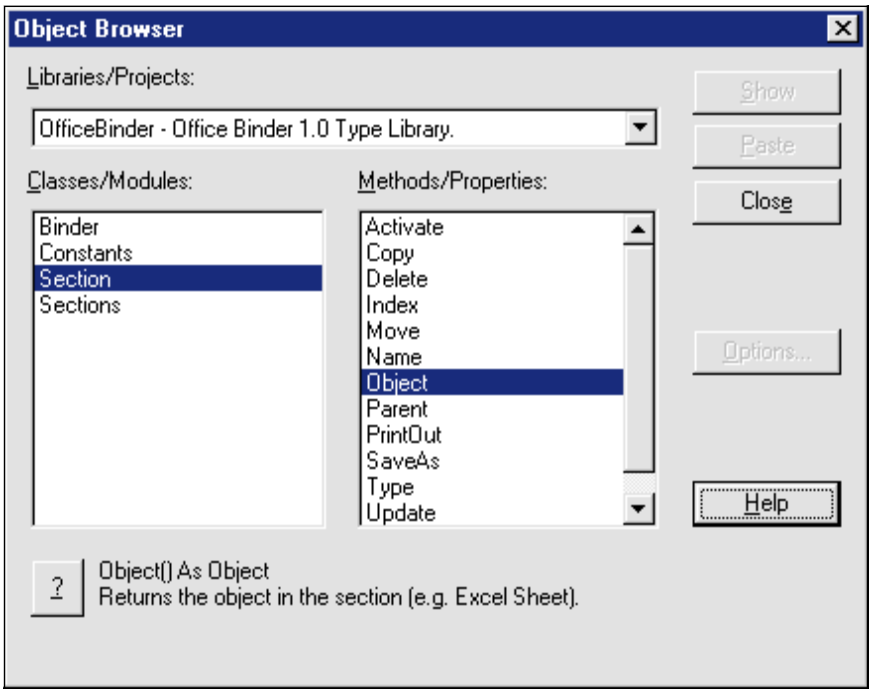

FIGURE 2 ***Automate OLE Objects in Your Web Browser.*** *The Microsoft OfficeBinder 1.0 type library exposes the Sections collection. Each Section object represents an embedded DocObject, which exposes as the Object property a pointer to the document's server; you use HTML-embedded Visual Basic Script code to manipulate the server's Application object. The object structure of Web pages you create with Internet Studio is expected to be modeled on Microsoft Binder.*

DocObjects are likely to be one of the primary mechanisms Microsoft uses to Internet-enable the next version of Office applications demonstrated by Bill Gates at the Internet Strategy Workshop. DocObject servers support OLE "abstract storage technology," which lets you save the data of multiple objects as a single entity; in the case of Binder, the persistent storage is a *FILENAME*.OBD, an Office Binder data file with an Office.Binder.95 association in Windows 95's Registry. (Binder also supports OBT template and OBZ wizard file extensions.) Abstract storage and moniker binders let DocObject containers activate linked files in place. Binder is an OLE Automation server with an OfficeBinder 1.0 type library to which you can create a reference in any VBA-enabled application, then create new or manipulate existing Binder Section objects (see Figure 2). Any non-trivial DocObject container is expected to expose its objects for manipulation through OLE Automation calls from VBA or VBS.

The Internet gets the most press attention, but enterprise-wide intranets are likely to generate the greater short-term revenue for Visual Basic developers, content designers/editors, and networking VARs. When you implement an intranet, users will expect to be able to use their browser to seamlessly view all of the documents to which they have access, not just documents that have been created in or converted to HTML. If you implement DocObject container features, you can provide a Web browser that not only displays Web pages in conventional HTML format via WinInet, but also lets the user browse documents created by DocObject servers that are stored as local or networked disk files through Windows' file functions. Instead of a link to a URL, you open the persistent storage file that holds the object's data from a folder. Depend-

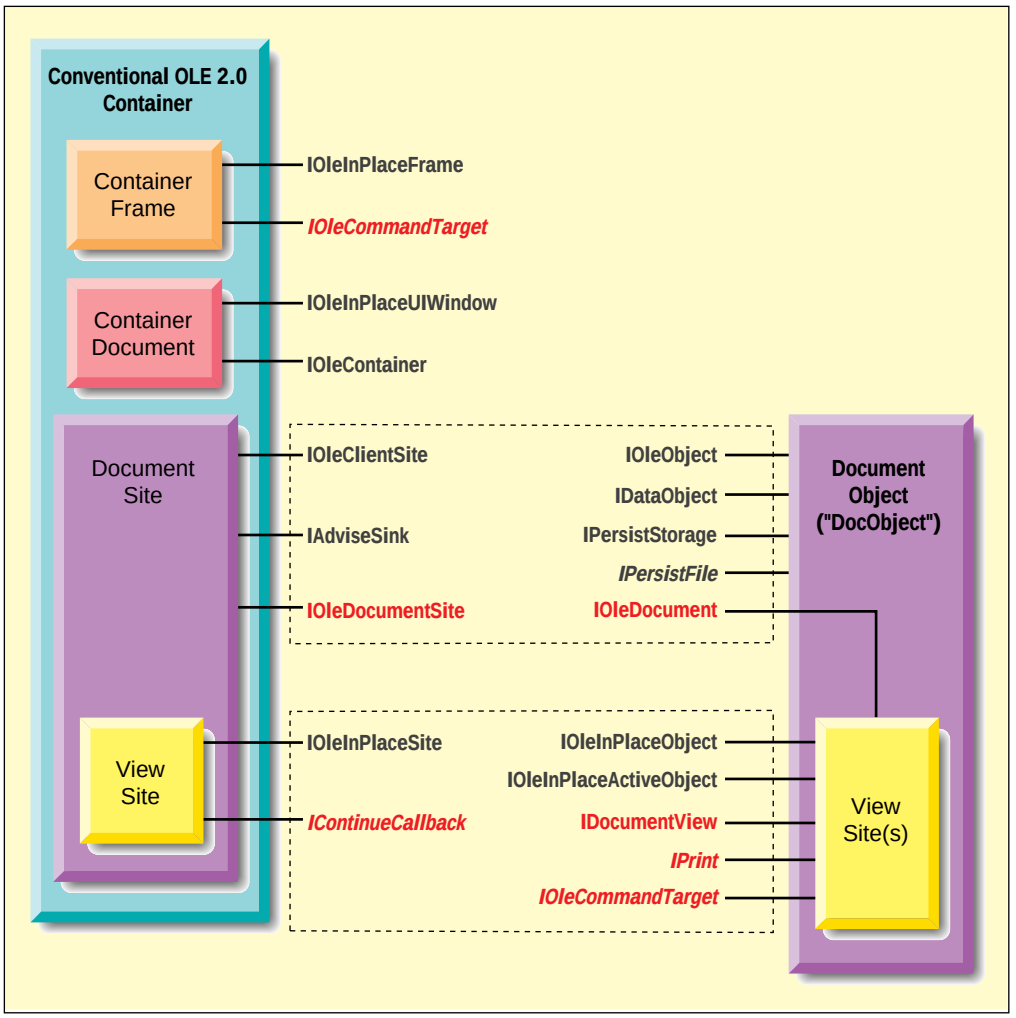

FIGURE 3 ***Just What You Wanted: More IOle... Interfaces.*** *DocObject containers for viewing DocObject-enabled documents must support at least the new IOleDocumentSite interface (red); IOleCommandTarget and IOleContinueCallback interfaces are optional (red italic). DocObject servers require implementation of IOleDocument and IOleDocumentView (red); IPrint and IOleCommandTarget are optional (red italic). Both the container and server must provide the standard OLE 2.0 interfaces for in-place activation. (This diagram is based on the early release of Microsoft's OLE Document Objects Specification, version 1.0.)*

ing on your file privileges or the version of the browser in use, you can view, modify, add, and/or delete the objects, including Web pages. Sharing an Excel worksheet for budgeting definitely beats filling in a conventional HTML form and using a Common Gateway Interface (CGI) script to handle the resulting data. A DocObject container with an Explorer-type interface would substitute a tree view of the local and network drives for the Section icon pane of Binder. (Bill Gates demonstrated a prototype DocObject container of the Explorer class at the Internet Strategy Workshop.) Another advantage of DocObjects is the ability to specify key words and other property values for searching; there's no equivalent of the Internet Information Server's Internet Fast Find feature in today's definition of HTML. Users won't need the suite of Office 95 apps to view embedded DocObjects; Microsoft promises a free DocObject-compatible viewer for viewing (but not editing) embedded documents.

Before its Internet Strategy Workshop, Microsoft published an early release of the specification for OLE Document Objects (version 1.0, November 27, 1995). You can obtain the current version of the specification as http://www.microsoft.com/intdev/inttech/docobj.htm. The specification enumerates the new IOle... and other I... interfaces that implement DocObjects: IOleDocument, IEnumOleDocumentViews, IOleDocumentView, IOleDocumentSite, IPrint, IContinueCallback, and IOleCommandTarget (see Figure 3). These interfaces supplement the existing IOle... interfaces associated with OLE 2.0, providing a means of manipulating server views within the container. Conventional OLE 2.0 containers have object, rather than document, views; thus printing documents that include headers, footers, footnotes, and other appurtenances presents difficulties. DocObject's IOleDocumentView, IPrint, and IContinueCallback interfaces are intended to solve such printing problems. DocObjects also support "menu merging," a more elegant method for grafting the server's menu structure to the container's menu bar. The Framer demo app, described in Kraig Brockschmidt's http://www.microsoft.com/intdev/inttech/framer.htm, implements the basic features of a DocObject container, but doesn't print the document.

According to Doug Heinrich, Microsoft's director of developer relations, Internet Explorer 3.0 will become a container for OLE objects, as well as conventional HTML documents. DocObjects obviously will play a major role in Microsoft's new browser, which Heinrich says should be available about the time you read this column. DocObjects reflect Microsoft's transition from conventional object orientation to the document centricity of the World Wide Web. Release of a preliminary but detailed specification for DocObjects is part of Microsoft's program to convince competing browser publishers to adopt OLE 2.x in their Windows offerings. The timing of publication of the DocObjects specification also may have been influenced by Apple Computer's announcement of a related approach to adding Internet connectivity to desktop apps through Cyberdog, which is based on OpenDoc object technology.

VISUAL BASIC SCRIPTS AND OLE SCRIPTING

Microsoft offers Visual Basic Script as an alternative to Sun Microsystems' nascent JavaScript for interactive Web-page authoring. Microsoft describes VBS as "a high-performance scripting language designed to create active, online content on

the World Wide Web." Microsoft intends to propose VBS to the World Wide Web Consortium (W3C) and the Internet Engineering Task Force (IETF) as "an open Internet scripting language standard" and will post a "source reference implementation" on the Internet. According to Microsoft's press release for VBS, there are more than 3 million developers creating Visual Basic and VBA applications. The claim that one in 50 of the purported 150 million users of Windows is a "developer" may be overstated, but there's no question that Visual Basic and VBA predominate in the Windows programming tools and productivity application scripting business. Clearly, Visual Basic developers and Microsoft Office power users will prefer programming interactive Web pages in a subset of VBA rather than creating their own objects with a new language, such as Java or JavaScript, derived from C++. Bristol Technology Inc. and MainSoft Corp. have announced their intention to port VBS to several flavors of Unix, and there's sure to be a version for Mac users, too. Thus there's little doubt that W3C and IETF ultimately will sanction both JavaScript and VBS in the form of Internet RFCs.

MICROSOFT MAY HAVE MISSED THE FIRST BOAT TO THE INTERNET, BUT THAT DOESN'T MEAN THAT BILL GATES AND CO. AREN'T UP TO SNUFF ON THE TECHNOLOGY SIDE.

OLE Scripting is Microsoft's mechanism for incorporating VBS and other scripts within DocObjects. A script is defined as "some executable blob" that can be source code in text format, compiled pseudo-code, or executable machine code. When a Web browser with OLE Scripting detects that a blob of script is incorporated in a compound Web document (called a project), the browser initializes the particular script engine, such as VBS or JavaScript, to process the script. The script engine is an in-process server (OLE DLL) and you must register it with a unique CLSID, plus its name in\HKEY_CLASSES_ROOT\ScriptEngine \<*Name*> in the Registry. After an initial interchange of data between the browser and the script engine, the engine starts and captures events triggered by controls (called projectitem objects) contained in the equivalent of a Visual Basic form (a class object). To implement this process, the script engine exposes IScripting (new) and IPersistStorage interfaces. IScriptingSite is implemented on the application (project) side to resolve the names of objects to their instances. The IGangConnectWithDefault interface is responsible for resolving the handling of events triggered by project item objects. OLESCRPT.HTM, available from http://www.microsoft.com/intdev/inttech, provides a preliminary description of the new OLE interfaces required to implement OLE Scripting.

Microsoft states that client-side VBS will enable Web page authors using Internet Studio "to link and automate a wide variety of objects in Web pages, including OLE objects and 'applets' created using the Java language." So, VBS will include features requried for OLE Automation programming. Microsoft's "Visual Basic Script: Visual Basic Comes to the Net" white paper (VBSINTRO.HTM, also available from http://www.microsoft.com/intdev/inttech) provides the following HTML example for a VBS source code blob that interacts with an OLE control and a Java applet embedded in a DocObject Web page:

```
<Insert>
   clsid = {"Insert class ID here"}
   OLEcontrol.Forecolor = True
   OLEcontrol.Animate
   JavaApplet.Forecolor = OLEcontrol.Forecolor
</Insert>
```

Achieving the objective of a "compact, high-performance" VBS requires considerable streamlining of the feature set of today's VBA implementation to reduce the size and complexity of the VBS interpreter. Based on Microsoft's preliminary "Visual Basic Script: Working Description" white paper of December 13, 1995 (VBSCRIPT.HTM), VBS doesn't offer all of the functionality of VBA, but it's reasonably certain that the utility of VBS within the HTML environment will at least match that of JavaScript. On the server side, you can choose between full-fledged Visual Basic and VBS for handling chores such as registering Web visitors in a database, delivering database content, and other heavier-duty operations, probably in conjunction with ISAPI. The advantage of ISAPI over CGI is that ISAPI functions offer improved performance by running in-process; CGI operations run in their own process space.

Web pages created with Internet Studio are OLE control containers, so you'll be able to incorporate 32-bit OCXs, in addition to DocObjects created by Office applications. The OLE Controls 96 specification (OCX96.HTM) adds several new features required to optimize OLE controls for use in Web pages. Sun Microsystems has announced a forthcoming Java OLE control that incorporates the Java runtime engine, allowing incorporation of Java applets within DocObjects for the Web. The Java OLE control, scheduled for release in the second quarter, eliminates the need to implement OLE Scripting, at least for Java code. (If your Registry is a mess now, just wait until you download and register for OLE Scripting a bunch of Java and/or OLE objects from each of your favorite Web sites.)

One by-product of the emergence of Java for creating downloadable applets may be broader acceptance of interpreted languages by the programming community as a whole. Assuming a runtime interpreter of reasonable size, downloading p-code for an applet is much more efficient than retrieving a self-contained executable file. The "Interpreted and Dynamic" chapter of Sun Microsystems', "The Java Language Environment: A White Paper," available as http://java.sun.com/whitePaper/java-whitepaper-7.html, makes Sun's case for interpreted, rather than compiled, programming languages. Visual Basic developers clamoring for a native Visual Basic compiler might want to read the chapter and then rethink their position, at least as to Internet-related applications.

December 1995 was a *very* interesting month for Visual Basic programmers, purveyors of Internet-related products, and investors in high-tech firms that depend on the Internet for the future earnings that justify today's high-flying stock prices. The first half of 1996 will determine if Microsoft's OLE- and VBS-based Internet initiative is destined for short-term success. Bill Gates said on December 7, 1995, "We're hard-core about the Internet. Anything we're focused on we're generally hard-core and we are focused on this and therefore, very hard-core." Microsoft's total dedication to OLE and big-time investment in OLE for the Internet ensure that OLE 2.x and VBS will be major components of future Web browsers. ×