IN THE UNITED STATES DISTRICT COURT
FOR THE DISTRICT OF COLUMBIA

STATE OF NEW YORK *ex. rel.* Attorney General ELIOT SPITZER, *et al.*, Plaintiffs, v. MICROSOFT CORPORATION, Defendant.	Civil Action No. 98-1233 (CKK)

Direct Testimony of Stuart E. Madnick

Submitted April 26, 2002

List of Figures

List of Tables

I. Introduction and Qualifications

1. My name is Stuart E. Madnick. I am the John Norris Maguire Professor of Information Technology at the Sloan School of Management and Professor of Engineering Systems in the Engineering Systems Division at the School of Engineering at the Massachusetts Institute of Technology (MIT). My curriculum vitae appears as Attachment A.

2. My areas of expertise relevant to the issues before the court in this case include principles of operating system design, software engineering, and their applications to businesses and other large organizations. My expertise in these areas has been gained through graduate training, research, teaching, consulting, and business experience, and is reflected in my over 250 papers and other publications.

3. Microsoft's counsel asked me to review the alternative remedies proposed in this litigation and to focus my attention on the following issues:

- The extent to which the alternative proposals balance the potential conflict between requiring disclosure of information to improve appropriate and commercially reasonable levels of "interoperability" and avoiding disclosure of information that competitors could use to replicate ("clone") features of Microsoft's products.
- The extent to which the alternative proposals are technically feasible to implement.
- Any additional provisions that raised significant issues from my technical perspective.

I am not an expert in antitrust law or economics. I do not offer any opinions as to the adequacy of either proposal in addressing the actions found to be anticompetitive by the Court of Appeals. My understanding, however, is that none of the acts found to be anticompetitive addressed issues of interoperability.

4. Section A provides more detailed information on my qualifications. Section B lists the steps I took in preparing the testimony. Section C provides a brief map of my testimony.

5. My testimony complements that of Professor John Bennett. For the most part, I have been asked to focus on different technical issues, although the complexity and interrelated nature of the issues raised by the non-settling States' Proposal may lead to some overlap in our respective testimony.

A. Qualifications

6. I have been a faculty member at MIT since 1972. I have also served as the head of MIT's Information Technologies Group for more than 20 years. During that time, the group has been rated as best in the nation among information technology programs in all major studies of which I am aware, including *U.S. News & World Reports*, *Business Week*, and *ComputerWorld*. I have also been a member of MIT's Center for eBusiness and MIT's Center for Transportation Studies, a member of the executive committee of MIT's Center for Information Systems Research, and an affiliate member of MIT's Laboratory for Computer Science.

7. I have extensive research and development experience in the fields of computer science and software engineering. I have been active for several decades in the design and development of computer software applications and operating systems. I hold degrees in Electrical Engineering (B.S. and M.S.), Management (M.S.), and Computer Science (Ph.D.) from MIT. In addition to being a member of the MIT faculty for nearly 30 years, I have been a visiting professor at Harvard University, Nanyang Technological University (Singapore), University of Newcastle (England), and the Technion (Israel). I am a member of several professional information technology, computing, and electrical engineering societies. I have

served in several leadership roles in these professional societies, including as a member of the Board of Governors of the Institute for Electrical and Electronics Engineers Computer Society and Chairman of its Technical Committee on Database Engineering.

8. I have researched and lectured extensively on subjects relating to computer operating systems, software development, and information technologies. Early in my career I was the principal author of a textbook, *Operating Systems* (McGraw-Hill 1974), which has been translated into several languages and which has been used in over 100 colleges and universities. I am also the co-author or editor of three other books: *Computer Security* (Academic Press 1979), *The Strategic Use of Information Technology* (Oxford University Press 1987) and *The Dynamics of Software Development* (Prentice-Hall 1991). The last book received an award from the Systems Dynamic Society in 1994 for "best contribution to the field of system dynamics in the preceding five years." In addition, I am the author or co-author of over 250 published articles, books, book chapters, and reports.

9. My current research interests include database technology, software project management, the strategic use of information technology, and interoperability among heterogeneous information systems running on multiple computers in networks. I am currently co-Director of MIT's PROductivity from Information Technology (PROFIT) initiative and co-Head of MIT's Total Data Quality Management (TDQM) research program. Until recently, I served as a member of the World Wide Web Consortium (W3C) XML Query Working Group on standards. I have served as Principal Investigator of a large-scale research effort funded by the Defense Advanced Research Projects Agency (DARPA) on Context Interchange, which studies technologies supporting the ability of organizations to work collaboratively. I have been awarded two software patents (with others) for work done as part of this effort. Current and

recent sponsors of my research at MIT include the US Air Force, Merrill-Lynch, MITRE, PricewaterhouseCoopers, Citibank, Fleet Bank, and Suruga Bank (Japan).

10. In addition to my research and development work in academia, I have extensive practical experience in the development of information systems for industry. I have been active in industry as a key designer and developer of projects including IBM's VM/370 virtual machine operating system, IBM's Script 370 word processing system, and Lockheed's DIALOG information retrieval system. I have also co-founded several high-tech firms, including Intercomp (acquired by Logicon), Mitrol (acquired by GE Information Systems), Cambridge Institute for Information Systems (predecessor to the Cambridge Technology Group and Cambridge Technology Partners, which is now owned by Novell Corporation), and iAggregate (acquired by ArsDigita, which in turn has been acquired by Red Hat).

11. I have also developed techniques for adapting software applications to run on platforms for which they were not originally designed, using tools that enable "porting" (modifying the program for the new platform) or using "emulation" (adding software or hardware to the new platform so that it emulates the one for which the application was designed). Most recently, I co-invented a patented technology called Automatic Web Wrapping. This technology allows programs that rely on the standard Open Data Base Connectivity (ODBC) API to obtain data from Web sites that use the rendering language and data transfer protocol, HyperText Mark-up Language/HyperText Transport Protocol (HTML/HTTP), respectively, which have become the backbone of Web communications.

12. I have conducted technical training programs for several organizations. For example, I trained several thousand AT&T technical representatives to use AT&T's version of the Unix operating system. (AT&T was the original developer of the Unix kernel, the foundation of many of the more than 20 variants of the Unix operating system.)

13. As a teacher in executive education programs, I have worked with hundreds of executives from major corporations during the past 25 years. In addition, I have served as a consultant to many corporations in recent years, including AT&T, Bankers Trust, British Telecom, Citibank, Deutsche Bank, FIAT, John Hancock, In-Q-It, Lockheed, Microsoft, MITRE, San Paulo Bank, Sequent Computer, and Teradyne. In 2001 I worked with Dr. Patrick Valduriez of INRIA-Paris analyzing allegations by Sun, Novell, and others concerning the interoperability of Microsoft's operating systems, and we submitted a report on the subject to the European Commission on Microsoft's behalf.

B. Preparation for Testimony

14. In forming my opinions in this case, I have reviewed the relevant portions of a wide range of materials, including portions of:

1. Microsoft's proposed remedy, which is the same as the Second Revised Proposed Final Judgment negotiated with the United States and the nine settling States, which I shall refer to as the "Proposed Settlement."
2. The non-settling States' Proposal, the most recent version of which is the Revised Proposed Final Judgment, which I shall refer to as the "non-settling States' Proposal."
3. Other legal documents on substantive issues submitted during this remedies phase, including the joint "Statement of Undisputed and Disputed Facts" and answers to interrogatories.
4. Selected legal documents from earlier phases in the case, including the District Court's Findings of Fact and the Court of Appeals decision of June 2001.
5. Reports from other experts, in particular, those of the other testifying computer scientists, Dr. Bennett and Dr. Appel.
6. Depositions of Dr. Bennett and Dr. Appel, Dr. Appel's written direct and oral testimony, and Dr. Shapiro's written testimony and cross examination.
7. Deposition of James Thomas Greene, Assistant Attorney General, California.

8. Testimony of certain Microsoft executives, including depositions of Brian Valentine, Rob Short, James Allchin, Ben Waldman, Roger Michael Needham, Linda Wolfe Averett, Bill Gates, and Steven A. Ballmer; and the direct testimony and cross examination of Bill Gates.

9. Testimony of certain Third Parties, including Carl Ledbetter, Richard Green, Michael Tiemann, Richard L. Ulmer, Scott Borduin, Steven McGeady, James Barksdale, David Richards, and Jonathan Schwartz.

10. Various discovery documents.

11. Relevant publications on specific issues, including articles in the trade press.

12. Various materials submitted to the European Commission by some of the same parties testifying in this case (*e.g.*, Sun and Novell), which I had reviewed earlier as part of my submission (with Professor Patrick Valduriez) to the Commission on issues related to interoperability.

13. The source code for Windows XP. (Microsoft provided me with the complete source code and binary files for Windows XP on a set of 10 CDs, which were installed on a computer for my inspection and use.)

Attachment B has a complete list of materials considered.

15. In preparing my testimony, I was assisted by research staff at National Economic Research Associates (NERA).

C. Organization of Testimony

The remainder of my testimony is organized into seven major sections:

- Section II provides some background on operating systems, including growth in features, interdependencies among the components of operating systems, and how operating systems provide services to software applications through application programming interfaces (APIs).
- Section III provides background on interoperability among different computers in networks. I understand that this subject was not addressed during the liability phase, but is covered by provisions in the remedies proposed by both Microsoft and the non-

settling States. I conclude that users are increasingly able to achieve interoperability among different applications and operating systems without the substantial disclosures and transfers of intellectual property that would be required by the non-settling States' Proposal. I emphasize that in analyzing this issue, it is important to distinguish "interoperability" from "interchangeability."

- Section IV analyzes several allegations made by Microsoft's competitors regarding interoperability issues, particularly as those allegations relate to client-server computer networks. I conclude that these complaints reflect an extreme position that conflates interoperability and interchangeability. The information they seek goes well beyond that which is needed for interoperability in the usual sense, and would, in addition, enable other firms to clone valuable functionality that Microsoft has developed.
- Section V addresses the information disclosure provisions (provisions 4, 12, 14 and 15) of the non-settling States' Proposal. Because the Proposal incorporates a sweeping yet "granular" definition of "Middleware," its disclosure requirements would have adverse effects even if they were truly restricted to interfaces. In fact, Microsoft would be required to disclose far more than the term "interface" suggests, thereby further facilitating the cloning of Microsoft product features by competitors.
- Section VI addresses the "unbinding" requirement (Provision 1) of the non-settling States' Proposal and concludes that it is technically infeasible as a consequence of the Proposal's definition of "Middleware" and the requirement that the code supporting functionality—not just the user interface—be removed.
- Section VII deals very briefly with two other provisions in the non-settling States' Proposal. Provision 5, "Notification of Knowing Interference with Performance," would impose a large burden on Microsoft and make it difficult to make changes to its operating systems. Provision 16, "Adherence to Industry Standards," is also unworkable as well as inconsistent with how such standards are created and followed by the industry.
- Section VIII summarizes my conclusions and contrasts the problematic nature of the non-settling States' Proposal with the more practical and measured approach taken to the same issues in the Proposed Settlement.

II. Operating Systems, Platforms, APIs, and Applications

16. This section provides background on operating systems and several software concepts that are important to understanding the impact of alternative remedies. In particular, I show how commercial operating systems have added new features over the years, many of which would be defined as "Middleware" under the non-settling States' Proposal.

A. The Evolution of Operating Systems

17. The functions performed by operating system software, particularly those that are designed to work with widely distributed hardware or to run general purpose applications (traits of the modern PC, for example), have grown dramatically in response to increasing hardware capabilities and expanded uses of computers. In many cases these new functions first appeared as separate software, sometimes sold separately from the operating system by third-party vendors. Over time, those functions have become increasingly integrated into operating systems themselves. As features are integrated into operating systems, they share code and become interdependent with other parts of the operating system. Often they expose application programming interfaces (APIs) and become essential resources not only for other parts of the operating system but also for third-party applications. Although the capabilities of operating systems vary widely, this general process of ever-increasing functionality has been universal, affecting computers of all types.

1. Early Computer Operating Systems Were Primitive.

18. The most basic function of an operating system is control of the hardware through management of memory and provision of interfaces between software applications and the system hardware.[1] The first computers did not have operating systems *per se*. Each program

[1] *See, e.g.*, Stuart E. Madnick and John J. Donovan, *Operating Systems* (New York: McGraw-Hill Book Company, 1974), Ch. 1.

(generally custom-written for that system) controlled the hardware directly. Early computer operating systems lacked basic features such as file systems (which store and organize data or applications on disks or other media) and the ability to run more than one application at a time.[2] Operating systems were unique to each computer model. The IBM System 360 series of the 1960s is generally credited as the first instance in which one operating system was used on a range of models of varying capabilities.

18. Operating systems for personal computers built on microprocessors followed a similar, though faster, evolutionary path. The first commercial personal computer, the Altair, did not have an operating system when it was introduced in 1975. When the IBM PC was introduced in 1981, microcomputer operating systems had been developed but were still relatively primitive. The original IBM PC could perform some functions without any operating system, and three operating systems (including PC-DOS, licensed by IBM from Microsoft) were available for use with it. None of them could access data stored on hard disks, and none had a built-in capacity to connect to other computers in networks. None required as much as 1 percent of the memory demanded by current operating systems.[3]

2. Features Have Been Added Steadily.

19. In the mid-1980s operating system vendors began adding mouse-driven Graphical User Interfaces (GUIs) to make computers easier to use. Microsoft Windows, which began to grow in popularity following the release of Version 3.0 in 1990, not only offered a GUI but also exposed its own APIs along with those in the underlying MS-DOS operating system. Windows also provided many additional operating system functions, such as memory

[2] They did not need either function because each program was submitted (often on punched cards) and executed by the computer separately.

[3] *The MS-DOS Encyclopedia*, Ray Duncan, ed., (Redmond, Washington: Microsoft Press, 1988), pp. 20–26.

management and the ability to run multiple programs simultaneously.[4] Over time Windows assumed more and more operating system functionality, relegating MS-DOS to a shrinking role, eventually serving as little more than the "boot loader" used to launch Windows when the PC is turned on.

20. During the early 1990s, Microsoft added substantial functionality to Windows, including scalable fonts that appeared the same on screen as on the printed page, multimedia extensions for playing audio and video files, and network client software to connect to servers or to other PCs in "peer-to-peer" networks.[5] In 1995, Microsoft introduced Windows 95, which combined and extended the functionality of MS-DOS and Windows 3.x.[6] In addition to Web-browsing software, Windows 95 provided the basic protocols for communicating over the Internet (TCP/IP) that were needed by many applications, notably Web browsing software. Windows 95 also included software for receiving and sending email.[7]

21. All other operating system vendors have also added substantial functionality to their products over time. This has been particularly true of microcomputer operating systems (including PCs), which were initially hampered by severe limitations on hardware resources. As hardware capabilities increased and both developers and users became more sophisticated, these more advanced operating systems expanded their capabilities.

---

[4] William S. Hall, "Windows 3.0 Under the Spotlight," *PC Magazine*, September 11, 1990, pp. 181–239.

[5] Gus Venditto, "Windows 3.1 Brings More of Everything," *PC Magazine*, April 28, 1992, pp. 123–160; Frank J. Derfler, Jr. and Steve Rigney, "Windows for Workgroups 3.11: The Best Windows for All?" *PC Magazine*, January 11, 1994, pp. 38–39.

[6] Michael J. Miller, "Getting Ready for Windows 95," *PC Magazine*, May 16, 1995, pp. 102–115.

[7] *See id.* "TCP/IP" stands for "Transmission Control Protocol" and "Internet Protocol."

[8] The Macintosh was preceded by Apple's Lisa computer, which also had a GUI, but was unsuccessful in the market. Xerox's Palo Alto Research Center (PARC) is usually credited with developing the basic GUI concepts in the 1970s. (*See* Owen W. Linzmayer, *Apple Confidential* (San Francisco, CA: No Starch Press, 1999), pp. 51-59; "PARC History," http://www.parc.xerox.com/hist-lst.html (downloaded March 22, 2002); "Xerox Alto Computer," http://members.fortunecity.com/pcmuseum/alto.html (downloaded March 22, 2002).)

22. New features emerge in many ways, sometimes as self-contained applications or utilities, other times as integrated parts of the OS. For example, in 1984 Apple's Macintosh operating system was the first commercially successful platform with a GUI. Unlike Windows, which was introduced in 1985 as a separate add-on to MS-DOS, the Macintosh GUI was always integrated into the operating system.[8] Similarly, OS/2—initially a joint effort by IBM and Microsoft, but developed and marketed exclusively by IBM starting in 1991—was, after 1988, sold only with its GUI.[9] IBM's OS/2 Warp 3 operating system included Web-browsing software (IBM's Web Explorer) in early 1995, more than six months before the release of Window 95 with Web browsing capabilities.[10]

23. Depending on the operating system and the specific feature, new features may be integrated into the operating system itself (*i.e.*, sharing code with other parts of the operating system and exposing APIs for third-party applications) or may be provided by stand-alone application software (sometimes licensed from third parties). For example, Windows, like all commercial operating systems of which I am aware, ships with a simple text editor (Notepad in the case of Windows) that is a relatively self-contained block of code that is easily removable.

24. Starting with the August 1996 service release of Windows 95 and continuing with Windows 98, Windows 98 SE, Windows ME, Windows 2000 Professional, and Windows XP on the desktop, Windows integrated Web-browsing functions with other parts of the operating system. The code that supports Web browsing also supports other operating system functions (including Help) and can be used by third-party applications.[11] KDE, one of the major

[9] Charles Petzold, "Why Windows 95? A Historical Overview," *PC Magazine*, January 24, 1995, pp. 255–257. Note that IBM was always the senior partner in OS/2, and after 1991 it was solely responsible for its development and sales.

[10] IBM, "OS/2 Warp: The Totally Cool Way to Run Your Computer," 1995 (DX 198).

[11] Microsoft Corporation's Response to Plaintiff Litigating States' First Set of Interrogatories in Remedy Proceedings, January 11, 2002, pp. 27-28.

GUIs used on the open-source Linux operating system, also integrates Web browsing using the Web browser as a file manager for files stored on the computer's own disk.[12] One cannot delete the Web browser from KDE without losing the ability to manage files on the user's own hard disk. GNOME, the other major Linux GUI, often ships with the Nautilus integrated file manager/Web browser.[13] Such integration serves several useful purposes: (1) the same code is used for multiple functions within the operating system, a design approach that economizes on system resources and makes updates easier; (2) users have a more consistent interface for common functions; and (3) third-party developers have reliable access to functionality exposed through APIs.

25. Table 1 compares some of the features included in six operating systems currently sold for personal computers: Windows XP, Mac OS X, IBM OS/2, Red Hat Linux, and two proprietary variants of Unix, IBM AIX and Sun Solaris.[14] Few of these features would have been part of any operating system 15 years ago, but most are now included in some fashion with all six. Although all of these features are explicitly defined by the non-settling States' Proposal as Middleware, or fall within its "Middleware" definition, Microsoft Windows XP is far from alone in providing them. Unlike Apple's Mac OS X and IBM's OS/2, Windows XP lacks voice recognition. Unlike OS/2 and the Red Hat distribution of Linux, Windows XP does not ship with a suite of business productivity applications. Like all of the other operating systems, Windows XP includes software for accessing email, Web browsing, and media playback. In each, the features shown are included with the operating system product without

[12] "Konqueror," http://www.konqueror.org/ (downloaded March 22, 2002).

[13] "GNOME 1.4 User's Guide: Nautilus User Manual," http://www.labs.redhat.com/gug/users-guide/nautilus.html (downloaded March 22, 2002).

[14] As discussed below, all of these operating systems are also sold in versions for computers used as servers in networks. The two commercial Unix variants (Solaris and AIX) are used primarily on servers, although some are sold with high-powered PCs used as workstations by engineers, scientists, and financial analysts.

Table 1. Comparison of Features Available in Current Operating Systems for PC Clients

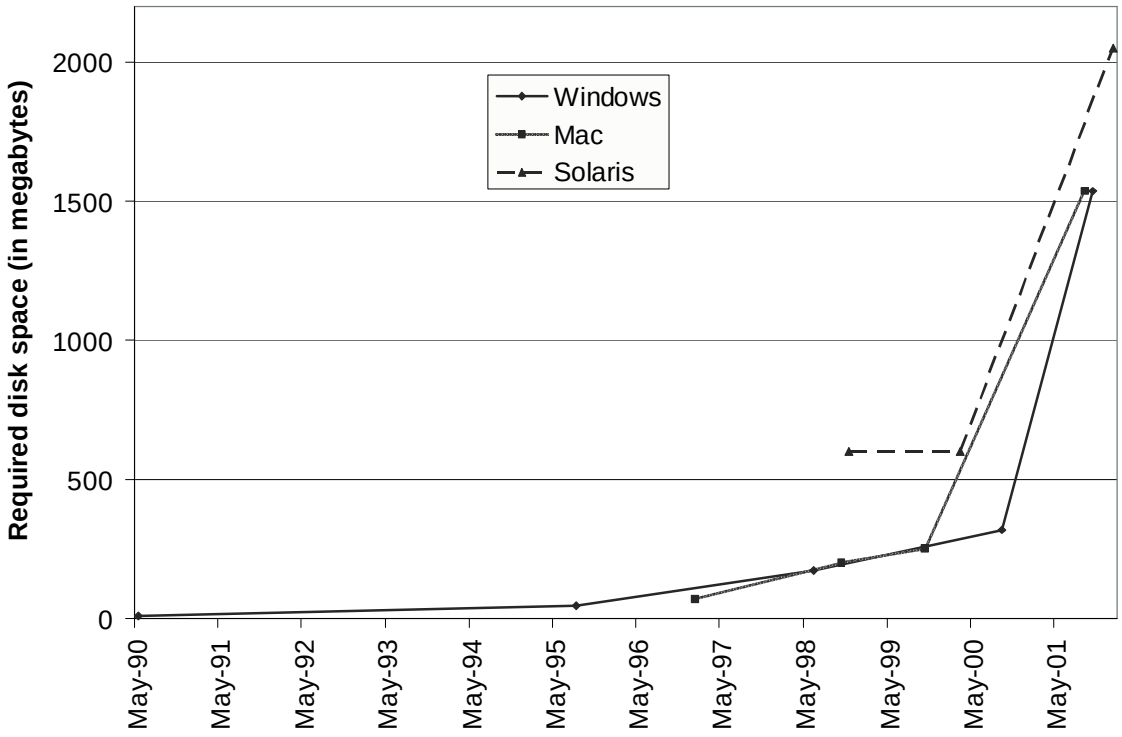
Feature	Microsoft Windows XP	Apple Mac OS X	IBM OS/2 Warp 4	IBM AIX	Red Hat Linux	Sun Solaris/ SPARC
TrueType fonts	■	■	■	■	■	■
Email client software	■	■	■	■	■	■
Graphical user interface	■	■	■	■	■	■
Network client/peer-to-peer networking	■	■	■	■	■	■
Web server	□[1]	■		■	■	■
"Calendaring systems"			■		■	■
Directory server				■	■	■
Email server				■	■	■
Voice recognition software		□[3]	■			
Java runtime environment	□[2]	■	□[4]	■	■	■
Media creation software	■	■	■	□[5]	■	■
Media playback software	■	■	■	□[5]	■	■
Instant messaging software	■				■	
Web browsing	■	■	■	■	■	■
Digital imaging software	■	■	■	□[5]	■	■
Handheld computing device synchronization software					■	■
Office suite			■		■	■
"Network operating systems"				■	■	■

Notes:
1) Web server included in Windows XP Professional but not in Windows XP Home.
2) Available as a Web download through "Install on Demand," or may be preinstalled on computer by OEM.
3) Commands only—no dictation.
4) Can run Java applications, but not applets on Web pages.
5) Included in AIX 4.3.3, may have been discontinued in AIX 5L.

Source: Neil Stokes et al., *Getting to Know OS/2 Warp 4* (Upper Saddle River, NJ: Prentice Hall PTR, 1996); Apple, IBM, Microsoft, Red Hat, and Sun Web sites.

separate charge; the buyer must purchase the whole package, but is free not to use all of the features. Indeed, in three cases (AIX, Mac OS X, and Solaris), the operating systems are sold as part of a complete system that includes the vendor's proprietary hardware. Microsoft, of course, does not manufacture personal computers and cannot offer that sort of integration.

26. The addition of functionality has led to rapid growth in both the amount of code required by operating systems and the computer resources needed to accommodate the system. Figure 1 shows how the sizes of operating systems from several vendors (Apple, Microsoft, and Sun) have increased. As the figure shows, all three operating system products have increased in size sharply over time, and all are currently of similar size.



Source: Apple Press Release, "Apple Announces Mac OS 7.6; First Milestone in New Operating System Strategy," January 7, 1997, http://product.info.apple.com/pr/press.releases/1997/q2/970107.pr.rel.macos76.html (downloaded February 22, 2002); "Mac OS 8.5: System Requirements for Install and Use," http://docs.info.apple.com/article.html?artnum=58073 (downloaded March 5, 2002); "Mac OS 9: System Requirements," http://docs.info.apple.com/article.html?artnum=60498 (downloaded March 5, 2002); "Apple - Mac OS X - How to Upgrade - Requirements," http://www.apple.com/macosx/upgrade/requirements.html (downloaded March 5, 2002); "Solaris 7 Operating Environment," http://www.sun.com/solaris/ds/ds-solaris7/solaris7.pdf (downloaded March 1, 2002); "The Solaris 8 Operating Environment," http://www.sun.com/solaris/ds/ds-sol8oe/ds-sol8oe.pdf (downloaded March 1, 2002); "Solaris 9 Operating Environment: Customer Early Access Program," http://www.sun.com/software/solaris/programs/solaris9ea/ (downloaded March 1, 2002); Original disks for DOS 4.0; "Disk Space Required for Windows 3.0 Installation," http://support.microsoft.com/default.aspx?scid=kb;en-us;Q63686 (downloaded April 9, 2002); "Windows 95 Installation Requirements," http://support.microsoft.com/default.aspx?scid=kb;en-us;Q138349 (downloaded March 5, 2002); "Minimum Hardware Requirements for a Windows 98 Installation," http://support.microsoft.com/default.aspx?scid=kb;en-us;Q182751 (downloaded March 1, 2002); "Minimum Hardware Requirements to Install Windows ME," http://support.microsoft.com/default.aspx?scid=kb;en-us;Q253695 (downloaded March 5, 2002); "System Requirements for Windows XP Operating Systems," http://support.microsoft.com/default.aspx?scid=kb;en-us;Q314865 (downloaded March 5, 2002).

Figure 1. With Increasing Functionality Have Come Increasingly Large Operating Systems

B. Operating Systems as Platforms for Applications

27. Operating systems provide some functions that users employ directly (*e.g.*, file managers), but their most important purpose is to provide a "platform" on which applications

can run and through which they can access the hardware. In this platform role, the operating system provides services that allow applications to work on the computer. Applications obtain most of these services through APIs. The richer the set of functions provided through APIs, the less effort the application developer must devote to writing code for basic functions, and the more the developer can focus on supporting higher level functions. As with all aspects of software design, however, there are tradeoffs—exposing APIs for public use is not costless—which I discuss more fully below, in Section V.B.

28. Applications running on MS-DOS and other early PC operating systems had to provide their own special software code called "printer drivers" if they wanted the printed output to include such features as bold text or italics.[15] Each application had to have a separate driver for each printer type it supported, because different printers offer different features and use different commands to control those features. The same was true for graphic video displays.

29. Windows and other modern operating systems greatly simplify the developer's task by providing an intermediate "abstraction" layer. The application sends a request for output to the operating system, which then has its own drivers for each printer or electronic video card supported. The application does not have to provide code for each type of printer or video card, only for the operating system(s) on which the application runs.

1. Exposing and Documenting APIs

30. Operating systems and other complex software contain many interfaces that, in effect, allow blocks of code to communicate with one another. Most of these interfaces are used only for internal purposes. APIs are interfaces that are "exposed" (made available for

[15] Printer drivers are specialized software that converts formatting information from an application to the commands needed by a specific printer. MS DOS provided printer support only for "generic text," which ignored the application's formatting and simply printed plain text. In 1990, WordPerfect 5.1 included support for 803 different printers. (*See* Edward Mendelson, "Word Processors: The Best and the Brightest," *PC Magazine*, December 11, 1990, pp. 107-192.)

applications to use) and "documented" (so that developers know how to use them). The platform vendor documents APIs by specifying what information must be provided by the software application using the API and what information (if any) is returned to the application. For example, a simple function for calculating square roots might specify the form in which the application presents the original number and how the result is returned (*e.g.*, what error code it will return if the input is negative, since it is not possible to take the square root of a negative number).[16]

31. An API for displaying a dialog box might require the application to specify the size of the box, its location on screen, and how options are to be presented inside the dialog box. Generally it would also specify how the application could determine the response(s) a user makes to the dialog box. By using such an API, the developer avoids having to write his own code for drawing the dialog box on the screen, thus saving effort and presenting the user with a more consistent set of dialog boxes and other visible interfaces. The testimony of Scott Borduin, the Chief Technology Officer at Autodesk, who has developed applications for several platforms, discusses the benefits that developers derive from a consistent platform with many APIs.[17]

2. APIs Do Not Include Implementation Details.

32. The details of precisely *how* the platform provides the service—going from the inputs supplied by the application to the outputs it desires—generally are not part of the documentation of the API. From the perspective of the developer using an API, that code is a

[16] Basic mathematical functions generally are provided by programming languages, not operating systems, but the principle is the same.

[17] *See generally* Direct Testimony of Scott Borduin, April 3, 2002.

"black box" or "abstraction."[18] There are two reasons for this standard approach. First, developers do not need the details to use the API, any more than I need to know whether a car I rent has four or six cylinders in order to drive it. If I want to go faster, I need to know to push harder on the accelerator, not all the steps that translate a change in accelerator pressure into greater speed. If I want to stop quickly, I need to know to push on the brake, not whether the brake creates friction with pads in the shape of drums or disks. Providing design drawings and internal specifications is more likely to confuse than inform me. As Scott Borduin of Autodesk testified, reliance on platform source code rather than the specifications of the APIs "is definitely not the preferred method" among ISVs seeking to make use of platform features, and the need for use of platform source code by ISVs generally reflects inadequate documentation of the APIs.[19]

33. The second reason for not disclosing the details of the code that implements APIs is that code embodies the intellectual property of the platform vendor, and intellectual property makes up the bulk of a software vendor's assets. If the vendor provided all of the details of how an API was implemented, it would be relatively easy for competitors to duplicate it. Even though the code is likely to be protected by copyright, access to the underlying source code and other details makes it relatively easy to imitate its functions without infringing the copyright by outright "copying."[20] That is why most commercial software companies protect

---

[18] In his direct testimony, Dr. Appel explains: "Abstraction is the hiding of low-level details; it is necessary (for example) because when an application wants to read a character from a file, it would like to do it the same way whether the file resides on the floppy disk, the hard disk, or the CD-ROM; the operating system can provide a uniform 'read a character' service, even though the different hardware devices each have different (and complicated) access mechanisms. Protection is necessary because an application program does not (usually) have carte blanche to operate all hardware devices: it can write to its own files, but is not allowed to write over the whole disk, is not allowed to display in other applications' windows on the screen, and so on." (*See* Direct Testimony of Dr. Andrew W. Appel, March 13, 2002, ¶ 26.)

[19] Trial Transcript, April 18, 2002, a.m. session (Borduin Cross), p. 4201-4202.

[20] A common use of the term "clone" is to refer to software that functions exactly like another product. Note that a "clone" duplicates functionality, not necessarily the precise code, and thus may escape copyright restrictions.

(continued...)

their source code by keeping it secret, sharing it only in limited circumstances and under tight disclosure rules. As I explain in detail below, the non-settling States' Proposal ignores both this convention of commercial software development and the relative ease with which competitors could "clone" Windows features given access to internal details of their implementation.

3. APIs and Componentization

34. Depending on how the software platform is designed, its APIs may expose functionalities at different levels, providing the developer with more or less detailed control of the characteristics of the service provided. The more discretely functions are exposed by the operating system, the more flexibility applications have to use them selectively and in a customized fashion.

35. The richer the functionality exposed by an operating system's APIs, the less code third-party developers have to write to obtain basic functions and the more effort they can devote to providing new features. For example, "toolbars" proved to be a very popular feature in Microsoft Excel. Microsoft added code to Windows and exposed APIs so that any application running on Windows could create its own customized toolbars relatively easily.[21] The developer can control, among many other things, the functions included on a toolbar, the graphical images (icons) used to identify the functions, and the location of the toolbar on the screen.

4. Windows Exposes Thousands of APIs.

36. Microsoft's Windows operating systems expose and document thousands of APIs. Developers can obtain the information needed to use those APIs (software development

(...continued)
(*See* "Clone," http://www.pcwebopedia.com/TERM/c/clone.html (downloaded March 21, 2002); "Clone," *Computer Desktop Encyclopedia*, 1999.)

[21] "Toolbar Controls Overview," http://msdn.microsoft.com/library/en-us/shellcc/platform/commctls/toolbar/toolbar_overview.asp?frame=true (downloaded April 8, 2002).

libraries and associated documentation) free of charge from Web sites maintained by Microsoft.[22] They can also buy software development tools (the Visual Studio line) from Microsoft that make it even easier to use the APIs.[23] For example, with some tools, typing in the name of a function may bring up a list of the required and optional information needed to use that function, with further information available by clicking a "Help" button or key. My understanding is that Microsoft does not maintain a count of the APIs in Windows; indeed, the precise definition of an API is ambiguous. In his testimony, Microsoft's founder, Bill Gates, put the number at more than 6,000.[24]

37. The availability of APIs on a platform plays an important role in the functionality that applications provide. For example, RealNetworks and AOL Time Warner both offer media players for Windows and the Macintosh.[25] In both cases, the most recent versions designed to run on Windows include Web browsing capabilities that make use of APIs that give developers access to the Web browsing code built into the Windows operating system. In contrast, the most recent versions of those media players running on the Macintosh do not have Web browsing capabilities because the Macintosh does not have similar APIs; the Web browsers that run on the Macintosh (including Microsoft's IE for the Macintosh and Netscape's

---

[22] *See, e.g.*, "Welcome to the MSDN Library," http://msdn.microsoft.com/library/ (downloaded January 21, 2002).

[23] *See, e.g.*, "Visual Studio Home Page," March 26, 2002, http://msdn.microsoft.com/vstudio/ (downloaded April 2, 2002).

[24] Direct Testimony of Bill Gates, April 18, 2002, ¶ 56. The Windows Platform software development kit (SDK) includes a file listing 12,416 programming "elements" in the "Win32 API." If one defines "API" more narrowly, the file lists just under 6,469 "functions" and "interfaces." (*See* "Win32API.csv," July 2001, http://www.microsoft.com/msdownload/platformsdk/sdkupdate/ (downloaded January 30, 2002).) Moreover, the Windows operating system includes additional documented APIs that are documented in SDKs other than the Windows Platform SDK.

[25] "Products & Services > Media Players," http://www.realnetworks.com/products/media_players.html (downloaded March 21, 2002); "RealOne Player," http://www.real.com/realone/sysreqs.html?src=011204404_1,020320realhome_2 (downloaded March 21, 2002); "AOL Time Warner | Shop & Subscribe," http://www.aoltimewarner.com/shop_subscribe.html (downloaded March 21, 2002); "Get Winamp Version 2.78," http://www.winamp.com/download (downloaded March 21, 2002); "Get Winamp | Mac Version .71 Alpha," http://www.winamp.com/download/mac (downloaded March 21, 2002).

Navigator) do not expose the necessary APIs. The richness of functionality provided by the Windows platform by giving API access to these components is one of its innovative and distinguishing characteristics.

38. It is important to note that developers do not have to rely on the operating system's APIs. For example, RealNetworks and AOL Time Warner could incorporate Web browsing code into the Macintosh versions of their media players. Adding such code, however, requires more work for the developer and increases the size of the files that need to be distributed, which may be problematic with types of software (such as media players) that are frequently downloaded from the Web at relatively slow transmission speeds.[26]

39. Similarly, the presence of APIs does not mean that applications have to use them to obtain the desired functionality. They can provide their own code. For example, if AOL or RealNetworks developed (or licensed) browser code for their media players on the Macintosh, they could modify ("port") that code for use in their Windows versions, rather than relying on Windows' built-in Web-browsing APIs. The Netscape and Opera Web browsers running on Windows, for example, provide their own code for rendering HTML (the standard language for presenting information on Web sites), rather than relying on the Windows APIs that provide access to the HTML rendering engine in the operating system.[27] Alternatively, applications may obtain services from software other than the operating system ("middleware," as the term

[26] Mr. Richards of RealNetworks testified that RealOne can use other browsing software, and thus would not be handicapped by the removal of IE code from Windows. (*See* Trial Transcript, March 20, 2002, a.m. session (Richards Cross), pp. 626-633.) I note, however, that RealNetworks has chosen to rely on that Windows code now, and does not have similar capabilities in its currently released software for other platforms, which suggests it finds the use of the APIs in Windows more efficient.

[27] "Netscape Gecko Developer Central," http://developer.netscape.com/tech/gecko/gecko.html (downloaded April 4, 2002) (Netscape 6.x's HTML rendering engine is called "Gecko."); "Opera's Secret: The Core," January 21, 2002, http://www.opera.com/docs/core/ (downloaded April 4, 2002).

has been used generally in this case) if that software exposes the necessary APIs. For example, Navigator could be rewritten to expose APIs for applications to use its HTML rendering code.

5. Windows APIs Support a Rich Array of Applications.

40. The APIs exposed by Windows support a rich array of applications—over 70,000 by one count cited by the Court of Appeals.[28] Only a small fraction of these applications were written by Microsoft.[29] Virtually none of the developers of the non-Microsoft applications had access to Windows source code. Indeed, my understanding is that Microsoft's policy is that even applications developed by Microsoft should rely only on documented APIs and should not use internal interfaces.[30] I would also point out that the most popular components of Microsoft's Office products—Word and Excel—have long been extremely popular on the Macintosh platform, and that Microsoft's developers did not have access to the source code for the Macintosh operating system in developing those applications.[31]

III. Interoperability In Computing Networks

41. Most computers today are connected to other computers in networks. These networks include both internal networks maintained by corporations and other large organizations ("enterprises" in the jargon of information technology) and the Internet (including the World Wide Web), to which any computer can connect with the appropriate hardware and software. My understanding is that interoperability among computers in networks

[28] United States Court of Appeals Decision, No. 00-5212, June 28, 2001, p. 55.

[29] For calendar year 2000, Microsoft applications and development tools revenues for all platforms was $9.9 billion; total applications and development tools revenues for 32-bit Windows platforms was $46.1 billion. (*See* IDC, *Worldwide Software Market Forecast Summary, 2001–2005*, Report #25569, September 2001, Tables 5, 8, 16, 17.)

[30] Deposition of James Allchin, February 13, 2002, pp. 236-238; Deposition of Henry Brian Valentine, January 29, 2002, pp. 36-41.

[31] Deposition of Ben Waldman, *Microsoft I-V Cases*, J.C.C.P. No. 4106 (Ca. Super.), August 22, 2001, pp. 489-491.

was not addressed during the liability phase of the case. However, both of the remedy proposals include provisions that address the issue. In this section, I provide some background on computer networks and on what interoperability means in practice.

A. Clients and Servers in Computer Networks

42. Most networks include both "client" and "server" computers.[32] The typical client device today is a desktop or laptop computer. But "thin-client" devices ranging from the Palm personal digital assistant (PDA) I carry in my pocket to "network client appliances" that look like PCs, but have limited capacities, are also components of many networks.

43. Servers provide computation, storage, and switching services to clients and to other servers. At the low end, specialized server "appliances" that cost less than $1,000 provide storage or other specific, limited services. At the next level, general purpose machines that use PC-style hardware and a single processor cost anywhere from $1,000 to several thousand dollars and provide a broader array of services. Larger networks often include more powerful servers with multiple processors and large amounts of storage that can cost anywhere from tens of thousands to millions of dollars. Many corporate networks include one or more traditional mainframe computers.

44. Most client computers are PCs that run some version of Windows. The operating systems used on servers and thin clients, however, are more heterogeneous, with none dominant except in particular niches. On thin clients, Windows CE (which is built on a separate code base from, and has far less functionality than, Windows XP or other Microsoft desktop operating systems) competes with the Palm OS, Linux, Symbian, and countless less

[32] Some networks are "peer-to-peer" and have no designated servers; users simply share the resources of their computers with other users in the network.

well-known operating systems, many of which are proprietary and designed to work with specific devices.[33]

45. On servers, there are many major vendors of operating systems. At the lower end of the price range, Windows, Linux, and Novell NetWare are the most popular server operating systems. At the high end, mainframes generally use proprietary operating systems (such as OS/390, used on IBM's S/390, now called the zSeries). Throughout the range of server system prices, though especially in the broad middle, proprietary variants of the Unix operating system (*e.g.*, Sun's Solaris, IBM's AIX, Hewlett Packard's HP-UX, Compaq's Tru64 Unix, or SGI's IRIX) are widely used, paired with proprietary hardware from the same vendor.[34] With the exception of proprietary operating systems used only on mainframes, however, there are no clear boundaries for server operating systems that are defined by system prices. Linux, for example, is used both on inexpensive server appliances and on multiple "virtual machines" running on IBM mainframes.[35] Similarly, Windows server operating systems are used not only

[33] In 2000, Wind River Systems led in shipments of embedded and handheld operating systems with a 22 percent share of units. Microsoft followed with a 21 percent share, Symbian had a 14 percent share, and Palm had a 6 percent share. (*See* IDC, *Embedded and Handheld Operating Environment Market Forecast and Analysis, 2001-2005*, Report #24862, 2001, Tables 2 and 3.) Also in 2000, 69 percent of handheld companions were shipped with Palm OS, 16 percent were shipped with Windows CE/Pocket PC, 5 percent were shipped with EPOC, and 0.3 percent were shipped with Linux. (*See* IDC, *Handcheck: The Smart Handheld Devices Market Forecast and Analysis, 2000-2005*, Report #24859, 2001, Table 12.)

[34] IDC's Server Tracker Database show that in 2000, 96 percent of all revenues from sales of server systems with NetWare installed were from systems selling for less than $25,000. The corresponding figures for Windows and Linux were 89 percent and 99 percent, respectively. In contrast, almost 100 percent (99.8 percent) of revenues from sales of systems with IBM's OS/390 were for systems costing more than $100,000. Unix sales (not including Linux) were more evenly divided, with 16 percent of sales from systems costing less than $25,000, and 38 percent costing less than $100,000. These figures understate the popularity of Linux for two reasons: (1) Linux is often installed on used hardware, which are not included in these figures, and (2) because Linux is open-source and available for free, revenue-based shares understate its importance. IDC estimates that of the number of new or upgraded server operating systems installed in 2000, 27 percent were Linux. (*See* IDC, *Worldwide Client and Server Operating Systems Market Forecast and Analysis Summary, 2001-2005*, Report #25118, August 2001, Table 2.)

[35] *See, e.g.*, "Sun Cobalt Qube 3 Server Appliance Specifications," http://www.sun.com/hardware/serverappliances/qube3/specs.html (downloaded March 22, 2002); "Sun Store U.S.: Sun Cobalt Qube 3 Server Appliance," http://store.sun.com/catalog/doc/BrowsePage.jhtml?catid=60312 (downloaded March 22, 2002); IBM Press Release, "IBM Introduces New Dedicated Linux Servers," January 25, 2002, http://www-1.ibm.com/servers/eserver/zseries/news/pressreleases/2002/zserieslinux_01-25-02.html (downloaded March 22, 2002).

on inexpensive servers, but also on multi-processor systems that cost hundreds of thousands of dollars.[36]

B. Interoperability

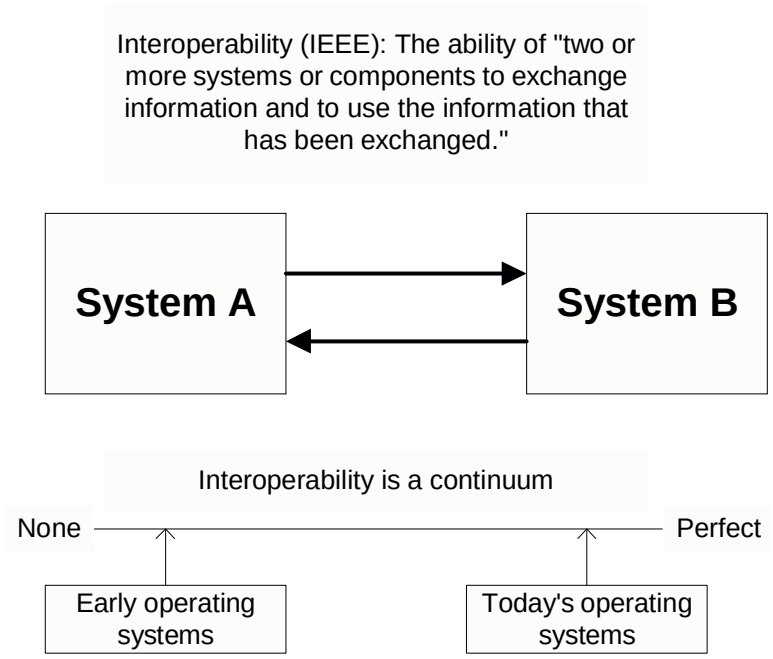
46. I agree with the witnesses for the non-settling States who have testified that, for networks to function, the various elements must "interoperate." I disagree, however, on both the way they use the term itself and on the sweeping conclusions they have sought to draw from this statement. Accepted definitions of "interoperate" are not consistent with the way the non-settling States' witnesses have used the term. For example, the Institute of Electrical and Electronics Engineers (IEEE), a well-respected organization of electrical and electronics engineers and scientists from industry, academia, and government, defines interoperability as "the ability of two or more systems or components to exchange information and to use the information that has been exchanged."[37] In my opinion, this definition is both reasonable and consistent with the use of the term in industry. I note that the concept of interoperability encompasses a continuum; it is not an absolute standard, as illustrated in Figure 2. Nor is there a single valid approach to interoperability.

47. The concept of interoperability applies to the full range of interactions in networks. For IT professionals in enterprises, the most vexing interoperability problems typically concern the ability of different applications to exchange information with one another in reasonably efficient ways when they were not originally designed to do so. Those kinds of interoperability problems arise whether the applications are running on one computer or

---

[36] *See, e.g.*, at the low end, "The Dell Online Store: Build Your System: PowerEdge 350," http://configure.us.dell.com/dellstore/config.asp?customer_id=04&keycode=6W300&order_code=PE350&cfgpg=3 (downloaded April 8, 2002). At the high end, see Deposition of Richard L. Ulmer, February 22, 2002, pp. 41–42, 99; Stephen Shankland, "Dell drops plan to sell Unisys server," *CNET News.com*, March 11, 2002, available at http://news.com.com/2100-1001-857174.html (downloaded April 8, 2002).

[37] Institute of Electrical and Electronics Engineers, *IEEE Standard Computer Dictionary: Compilation of IEEE Standard Computer Glossaries* (New York, NY: 1990), p. 114.



Source: Definition from Institute of Electrical and Electronics Engineers, *IEEE Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries* (New York, NY: 1990)

Figure 2. Interoperability: Definition and Continuum

several—and, if the latter, whether the computers use the same or different hardware and operating systems. In fact, the interoperability issues that are of primary concern to companies rarely involve the issues being raised by the non-settling States. For example, Citibank is expected to spend between $100 million and $500 million and four years to replace its existing Cosmos systems to improve interoperability—all of the software involved was developed by Citibank or its contractors.[38]

48. My understanding is that one focus of the non-settling States in this phase of the case is client-server interoperability—*i.e.*, the ability to connect clients to servers so that they

[38] Lucas Mearian, "Citibank Moves to Swap Out Aging Back-office Banking Platform," *Computerworld*, January 25, 2002, available at http://www.computerworld.com/cwi/Printer_Friendly_Version/0,1212,KEY52_STO67733-,00.html (downloaded April 8, 2002) ("'As the bank grew, we did make a mistake in that we released the source code [Cosmos] to each of the countries, and they changed it. Now we've ended up here 30 years later,' Berg said. By 'here,' Berg is referring to a platform that is proprietary to each country it's in and uses that nation's language, regulatory rules and business processes—something Citibank can no longer do in a global economy.").

can "talk" to each other and share information and resources such as printers and various storage devices. In particular, because of the predominance of Windows on PCs used as clients, the focus is on interoperability between Windows clients and non-Windows servers. I address the context in which to analyze interoperability issues below.

1. Interoperability Has Grown Over Time, and Heterogeneous Networks Are Common.

49. The non-settling States' Proposal implies that Microsoft's actions have reduced the ability of computers to interoperate in networks. In fact, interoperability has grown dramatically over time—especially in the past decade—as corporations and other organizations increasingly have demanded ways to make heterogeneous computers work together in performing complex tasks.

50. When mainframes were the primary type of computer in use, many companies chose one vendor wherever possible in order to minimize interoperability issues. One company might be known as an "IBM shop," another as a "DEC shop." The client devices were generally "dumb" terminals (*i.e.*, they did not have computing capabilities of their own) that were developed and sold by the mainframe vendor.

51. As additional types of computers were introduced, the strategy of standardizing on one type of computer broke down. Initially, as different types of computers were added, they were grouped into relatively homogeneous networks, where each type of computer communicated only with other computers of its type or with terminals and other specialized client devices. Communication among these networks was often minimal or nonexistent.

52. Today, most sizable private and public organizations have interconnected these various types of computers or sub-networks of computers into larger networks that span the organization (and often go beyond it). The computers in these "enterprise" networks typically range from hand-held devices to PCs to large servers. They range in age from the latest

technology to "legacy" machines built a decade or more ago. They use a variety of operating systems, everything from open-source software (such as Linux), to commercial operating systems that run on hardware from many vendors (such as various versions of Microsoft Windows and Novell's NetWare), to proprietary systems that are unique to a particular vendor's family of systems (such as IBM's OS/390 and the Apple Mac OS).

53. Surveys show that heterogeneous networks are the norm, not the exception. For example, IDC reported in 2000 that a survey of European firms with networks showed "overall 70% of companies interviewed have a multiple server operating environment. This percentage is even higher among major companies," 90 percent of which reported heterogeneous networks.[39] A recent survey conducted by Mercer Management for submission to the European Commission on Microsoft's behalf showed similar results. In a sample of European and U.S. firms with more than 500 client computers, almost 90 percent had a heterogeneous mix of servers.[40] Although Windows operating systems account for the vast majority of PC clients, companies use a wide mix of server operating systems, with the vast majority employing two or more different server operating systems.

54. My own university, MIT, provides an example. MIT is a moderate-size single-location organization of about 1,000 faculty and 9,000 students, but has thousands of servers of almost every type imaginable. The number and types of servers, scattered across the campus, change so frequently that no precise count or breakdown exists. I was able to determine that within my department, the Sloan School of Management, we have about 50 of our servers in a

[39] IDC, *TIPS Results: Server Operating Environments, Storage Systems and Enterprise Solutions*, Report #MTO4G, August 2000, pp. 2, 6.

[40] Mercer Management Consulting, "Managing Interoperability in Heterogeneous IT Environments: Primary Customer Research Findings," December 2001, p. 6.

single basement room running a mix of server operating systems, the primary ones being Linux, Sun Microsystems' Solaris, and Windows.

C. Levels of Interoperability: Tight vs. Loose Coupling

55. The concepts of "tight" and "loose" coupling provide a useful framework for thinking about degrees of interoperability. These concepts are commonly used in discussion of interoperability among different database management software systems. Such software products provide the platforms for many enterprise applications, most of which involve extracting, processing, and analyzing large data sets. These products provide APIs (interface functions) that developers use to write custom database applications (such as systems for tracking inventory and maintaining information on bank account balances).

1. An Analogy: Asking and Answering a Question

56. As an analogy, let us consider interactions between two people. Some interactions are sufficiently simple and standardized that it makes little difference whether I am talking with a close colleague or a complete stranger. If either asks me: "What time is it?" I would understand that question and they would understand a reply such as: "10:30 AM."

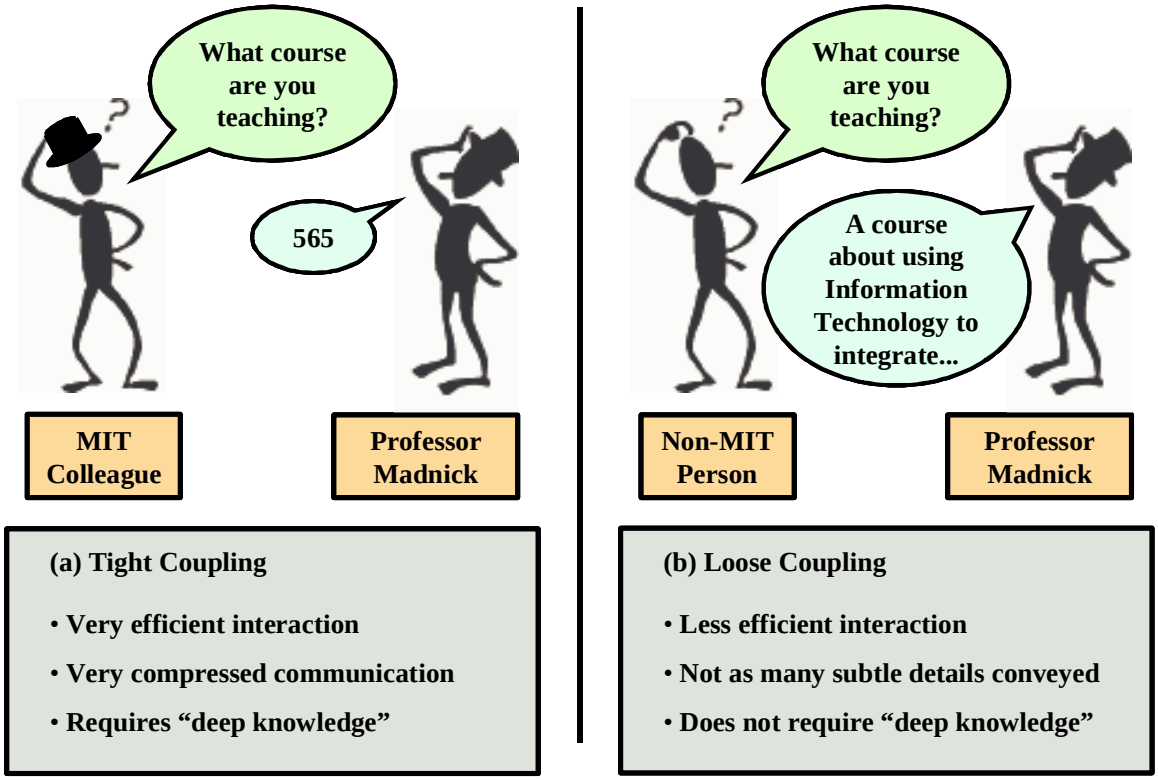


Figure 3. An Example of Tight and Loose Coupling

57. To better understand "loose" and "tight" coupling, consider another analogy, illustrated in Figure 3. Suppose I am asked: "what are you teaching this semester?" How I answer depends on who is asking, what they know, and what information they need. If it is a former student or departmental colleague, I can simply say "565," and they will know that I am referring to MIT course 15.565, where "15" refers to courses offered by the Sloan School in MIT and "565" refers to the specific course, "Integrating Information Systems: Technology, Strategy, and Organizational Factors." They also will know the basic content of the course and how it is taught. If they are interested in finer details, I can simply update the information I know they already have. For example, I may say that I have dropped the segment on topic X or that enrollment is up from last year. We communicate quickly and accurately with a minimum of information exchanged. That is "tight coupling."

58. Now suppose the questioner is from outside MIT. Saying "565" is unlikely to mean anything. At this point, I have several choices. I can tell them everything I can think of about the course, but that is unlikely to be a good use of time for either of us. If I know something about the questioner's background, I can tailor my answer to what I think they are most interested in and able to understand. Or, I can say something very general ("I'm teaching a course about using Information Technology to integrate …") and let them ask more specific questions. The less I know about them and the less we have in common, the more "loosely coupled" our conversation is likely to be and the more we will have to make detours to provide the context necessary to understand my answers. To have an in-depth exchange—which I can convey to a colleague by saying "565"—we would have to spend considerable time finding a common vocabulary and points of reference.

59. Tightly coupled interactions are much subtler, relying on shorthand based on common experiences and a deep knowledge of the field—which, in the example of Figure 3, is especially challenging because course numbers, numbering conventions, and course contents vary widely among universities. These same factors often occur in software systems, which often differ in their basic architectures and almost always vary in details of implementation.

60. Tight coupling of database management systems from different vendors has long been the goal of much academic and commercial research. The theoretical promise of tightly coupled databases is that applications could view them as a single global database without having to worry about dealing with different interfaces. In practice, however, tight coupling has been extremely difficult to achieve. There are simply too many details that must be specified and too many ways of accomplishing similar tasks. As a result, in most cases different databases exchange information via looser methods, such as custom queries from one database

to the other or periodic copying of the data, rather than trying to have both work from each other's portion of the database.

D. Customers Achieve Client-Server Interoperability in Many Ways.

61. Interactions among computer systems also may be classified in terms of how "tightly" they interoperate. For two computers to interoperate, they must have compatible software installed. Interoperability is easiest and most complete (tightly coupled) when both computers are from the same hardware-software platform family. In that case, they automatically share common points of reference (*e.g.*, file systems and interfaces), so that precise communication can be accomplished with a minimum of information changing hands. Like my colleague and me in the "tight coupling" example above, a Sun workstation and a Sun server, both running Solaris on SPARC processors, share a common, tightly coupled "vocabulary" when they work with one another, as do a Windows XP Professional PC and a Windows 2000 server.

62. Nonetheless, vendors and customers have found many ways to achieve satisfactory levels of interoperability in the mixed networks that predominate in most organizations. Many of the most sophisticated and tightly coupled solutions address the common situation of an Intel-compatible PC client running Windows connected to servers running non-Microsoft operating systems on non-Intel hardware.

1. Interoperability Among Computers with the Same Operating Systems

63. With the exception of Novell NetWare and certain proprietary operating systems used on mainframes and minicomputers, all of the operating systems (and most of the hardware components) used on servers are also used on client computers. Thus, Sun's Solaris operating system and its SPARC microprocessors are used on both servers and workstations sold by Sun. Similarly, IBM, Hewlett-Packard, Compaq, and SGI sell both server and client systems that use

their own variants of Unix.[41] Linux, the popular open-source variant of Unix, is used on both clients and servers. Apple sells client and server versions of its Mac OS X operating system, which run on Apple's Macintosh hardware.[42] Microsoft's current server operating system, Windows 2000, is also offered in a client version (Windows 2000 Professional, the successor to which is Windows XP Professional), and both run on Intel-compatible processors.

64. In all of these examples, the client and server versions of the operating systems are built on a common base of code, with a very high degree of overlap. Applications that run on the client version generally will run on the server version of the same operating system.[43] In every case of which I am aware, the client version of the operating system comes with software built-in that enables the client computer to connect to a server running the same operating system. In addition, because the various flavors of Unix share a common heritage, usually a client running one variant can connect relatively easily to a sever running another variant. Even in that case, however, the functions available to the client may be different, depending on the software included with the server. For example, Sun ships its iPlanet Directory Server software with its Solaris server operating system. A Sun client computer connected to an IBM server running AIX will not have access to iPlanet's directory services unless the company buys a

[41] When Unix was first used on computers with microprocessors by Sun and other vendors in the 1980s, virtually all of the systems were individual workstations used as substitutes for mainframes in computation-intensive tasks. Over time, however, most Unix systems (at least on a dollar volume basis) have become servers.

[42] "Step 2: Shop for Software," http://store.apple.com/1-800-MY-APPLE/WebObjects/AppleStore.woa/93/wo/491LT0ZTwAME8Zk1KM/0.3.0.3.30.5.0.CoverPageLeftSoftwarePromo.1.1.0.3.0.11.3.1.1.0 (downloaded April 1, 2002).

[43] The reverse is less likely to be true, because the server versions of the operating system often include additional functionality not present (or not activated) in the client. Note that for operating systems that run on different hardware, applications written to run on an operating system on one type of processor will require modification (at least recompilation into machine code) to run on the same operating system but a different processor. For example, there are many applications for Sun's Solaris running on its SPARC chips that are not available for the version of Solaris running on Intel hardware; examples include Sun's iPlanet Directory Server and its iPlanet Application Server. (*See* "iPlanet Directory Server Overview," http://www.iplanet.com/products/iplanet_directory/home_directory.html (downloaded April 4, 2002); "Downloads: iPlanet Application Server 6.5," http://www.iplanet.com/downloads/download/5208.html (downloaded April 4, 2002).)

separate copy of the iPlanet product. Even then, iPlanet Directory Server includes some features (the Solaris Extensions for iPlanet Directory Server, which provide "native support of the NIS and RADIUS protocols") that work only on Sun servers running Solaris.[44] In my experience, it is common for software products to be able to offer additional features when paired with other products from the same vendor that have been designed to work together. There are both technical and valid business reasons for this phenomenon.

2. Connecting Different Computers Running Different Platforms

65. Connecting client computers to servers using different hardware and operating systems requires the use of "loose coupling" methods or the installation of additional software on the client or server computers. Basic differences among operating systems mean that a computer cannot work in exactly the same way with a computer running a different operating system as it would with one running the same operating system. For example, the Mac OS, NetWare, variants of Unix, and Windows all store files in different ways and track different file characteristics.

66. One simple example is that Unix operating systems normally treat upper- and lower-case letters as different characters in file names: "Madnick" is a different file than "MADNICK," "madnick," or "MadnicK." But non-Unix operating systems generally do not distinguish file names on the basis of case. All of these distinct Unix file names are the same name on most non-Unix operating systems. Thus, sequentially copying these four separate files from a Unix system to a non-Unix system would result in a single file, the last one copied; the first three would be overwritten in succession on the non-Unix system.

[44] The server must also use Sun's SPARC chips; these features are not available on servers running the version of Solaris that runs on Intel processors. (*See* "Solaris Extensions for iPlanet Directory Server Data Sheet," http://www.iplanet.com/products/solaris_extensions/ds_solaris_pfv.html (downloaded April 2, 2002).)

67. Overcoming differences between client and server capabilities generally requires either the use of loose coupling methods or the installation of additional software on the client, the server, or both. Often these solutions involve finding some sort of least common denominator. For example, it is much easier to have a Unix system ignore the case of the letters in a file name than it is to modify other operating systems to distinguish file names based on the case on individual characters. In the process of finding common ground between different systems, however, some features unique to the individual systems may be lost. For example, by changing the capitalization of the letters in my name, $2^7 = 128$ different file names are possible, whereas only one is available if case is ignored.[45]

68. Getting different types of computer systems to "talk" to one another is a little like facilitating communication between people who do not share a native tongue. Suppose I am the "client" trying to exchange information with an airline agent (the "server") in Italy. My native language is English, while the server's is Italian. There are several possible ways for us to communicate, none of which will be as smooth and precise as the communication either one of us would expect with someone who spoke our own language perfectly fluently:

1. We can identify a common second language that we both speak. If we both know French, we can communicate in that language, though neither of us has the same facility with this third language that we have with our respective native languages.
2. We can hire a translator who knows English and Italian, and can serve as the "gateway" through which we communicate.
3. The server can learn some English, possibly focusing on the vocabulary most relevant to the clients.

[45] Of course, what some Unix users may regard as a "feature," others may regard as a source of confusion—they would prefer not to have to worry about the case of individual characters in file names.

4. I can learn some Italian, again focusing on the vocabulary most relevant to asking questions and understanding replies from the server.

As I discuss below, each of these types of approaches is used by enterprises to connect Windows clients to non-Windows servers.

a. Internet Protocols: A Common "Second Language"

69. The growth of the Web has spurred both demand and capabilities for greater interoperability. Ten years ago, servers running some variant of Unix generally used TCP/IP as their basic networking protocols, but NetWare, Windows, and Macintosh servers all used proprietary protocols of their own. Today, all of these server operating systems (plus Linux, which was virtually unknown 10 years ago) provide "native" support for TCP/IP, which is used on both internal networks and for connection to the Internet.

70. Many companies are turning to Web-type interfaces to achieve interoperability, both over the Internet and on internal organization networks. Many newer server applications, for example, rely on a Web browser interface.[46] Typically it does not matter whether the Web browser running on the client computer is Netscape Navigator, Microsoft's Internet Explorer, or some other Web browsing software, so long as it is a version of such software that supports key Internet standards. Nor does it matter whether the particular Web browser is running on Windows, Linux, Macintosh, or a commercial variant of Unix. Similarly, the client software does not care what kind of hardware or software the server is running, so long as it can communicate using the HyperText Transfer Protocol (HTTP), a simple protocol that builds on the ubiquitous TCP/IP. With HTTP and HTML, the Web has enabled "universal" client-server computing, so that a client can have uniform access to any Web server inside or outside the

[46] *See, e.g.*, Bob Trott, "PeopleSoft Makes Pure-Web Play in CRM," *InfoWorld*, May 30, 2001, available at http://www.itworld.com/App/651/PeopleSoftmakespure-215/pfindex.html (downloaded April 8, 2002); Renee Boucher Ferguson, "Oracle 11i Still Bugging Users," *eWEEK*, February 18, 2002, pp. 1, 15.

enterprise. Surveys of corporate IT departments by IDC show that almost 80 percent of respondents reported using Web browser access to corporate applications in December 2000, up from 63 percent just a year earlier.[47]

71. Using such techniques, from the PC on my desk at home or work, I am able to access information on hundreds of servers at MIT and millions of servers around the world—an extraordinary degree of information accessibility and interoperability.

b. Gateway Servers: Using a Translator

72. Another approach to achieving commercially reasonable degrees of interoperability is to connect Windows clients to a Windows server that provides a gateway to non-Microsoft servers. In such a configuration, the clients talk to the Windows server in their "native" language. Using software obtained from Microsoft (SNA Server, now succeeded by Host Integration Server[48]) or various third parties (for example, e-Vantage SNA Gateway from Attachmate[49]), the Windows gateway server passes on information requests from the Windows clients to the non-Windows servers. Only the gateway needs to be "bilingual."

c. Adding Software to the Server: Servers Learn To Speak "Client"

73. Many organizations that wish to connect Windows clients to Unix or Linux servers install software on the server that gives it the appearance of a Windows server when dealing with Windows clients. Some organizations use software from Unix vendors (*e.g.*, Sun's

[47] IDC, *Enterprise Technology Trends: Customer Directions and Vertical Market Opportunities - TIPS 2000 in Review*, Report #24327, April 2001, Figure 5.

[48] "Microsoft Host Integration Server—Previous Products," http://www.microsoft.com/hiserver/evaluation/previousversions/ (downloaded April 2, 2002).

[49] "Attachmate e-Vantage SNA Gateway Version 2.1," http://www.attachmate.com/products/printer_friendly/0,1743,252_1,00.html (downloaded April 9, 2002).

PC NetLink, which Sun includes free of charge with its Solaris server operating system), while many use the open-source software developed by an organization known as Samba.[50]

74. Such software lets the Unix or Linux server impersonate a Windows server in many important respects, so that it can provide file and print services to Windows clients in the same way that Windows servers do. Samba and its proprietary counterparts have cloned Microsoft's features sufficiently closely that they, too, are able to provide most network functions in a manner that lets Windows clients communicate as if they were dealing with a server running Microsoft's Windows NT 4 server operating system. As a result, it is possible to have a network of Windows clients using Microsoft's networking protocols, but with all of the servers running non-Microsoft operating systems on non-Intel hardware.

75. Microsoft's competitors and the open-source movement have not (yet) been able to clone some of the functions added to Microsoft's latest server operating system, Windows 2000 Server. That inability to duplicate all new functionality underlies most of the complaints competitors have made about "interoperability." However, as I discuss below in greater detail, Samba and similar software permit non-Windows servers to play important roles even in networks in which the customer has chosen to employ the Active Directory component of Windows 2000 to organize the network (sometimes known as running in Windows 2000 "native mode"). Moreover, in light of the progress made by the Samba team in the past, they may well succeed in the future in "cloning" the new features.

d. Adding Software to the Client: Clients Learn To Speak "Server"

76. It is easy from a technical perspective for Microsoft's competitors to teach Windows clients how to speak the competing server's "language" by installing some of their

[50] Although PC NetLink was developed using an AT&T product based on source code licensed from Microsoft (a license no longer in force), Samba was developed without access to source code.

own software on the client system. That is because the server vendor knows the details of its own server operating system, so it knows the precise "vocabulary" it needs to build into the client software. Like any application running on Windows, that client software can use published Windows APIs to work smoothly with Windows client operating systems. With such client software, Windows clients generally are able to make use of all of the services offered by non-Microsoft servers. In addition, such Windows client software generally can be installed automatically from the server using products from Microsoft or others (such as Novell's ZENWorks). Similar software lets non-Windows clients work with Windows servers.[51]

77. Because Novell's NetWare server operating system does not have a client counterpart, Novell has always included software to be installed on the client computer with NetWare. Starting with Windows for Workgroups 3.1 in 1992, Windows operating systems have included client software that provided connections to NetWare servers.[52] The client software originally shipped with Windows was developed by Microsoft and was based on public specifications and reverse engineering work undertaken by Microsoft, not access to NetWare source code or Novell's proprietary communication protocols.[53] More recently, Microsoft also included Novell's client software on the CD-ROM versions of Windows—as separate software, not a part of Windows.[54] Novell has always encouraged customers to use its full client software, which it includes with copies of NetWare and also makes available for

[51] "Citrix MetaFrame XPe: Product Brief," http://download2.citrix.com/ctxlibrary/products/pdf/XPe_prodbrief.pdf (downloaded April 23, 2002); "What's Samba all about?" http://us1.samba.org/samba/about.html (downloaded April 23, 2002).

[52] "Windows for Workgroups Version History (Q126746)," December 7, 2000, http://support.microsoft.com/default.aspx?scid=kb;EN-US;q126746 (downloaded April 11, 2002).

[53] "Reverse engineering" means using publicly available data and testing to determine how to replicate proprietary functionality without access to proprietary information (such as source code). Most open-source software involves such reverse engineering—providing functionality that "clones" commercial products without gaining access to the source code or violating intellectual property protection. Commercial software companies also do such reverse engineering.

[54] Direct Testimony of Dr. Carl S. Ledbetter, March 27, 2002, ¶ 117.

download.[55] In the newest version of NetWare, Windows clients can connect without the addition of NetWare-specific software. But, as discussed below, special client software is still needed to obtain all of the functionality that NetWare offers.

78. Similar solutions are available to connect Windows clients to Unix or Linux servers. Sun, for example, used to sell Solstice for use on Windows clients connected to Sun's Solaris servers.[56] WRQ now offers Reflection NFS Client for connecting Windows clients to Unix servers using the Network File System (NFS), which is common on servers running Solaris and other flavors of Unix.[57] It also offers six other Reflection products for connecting Windows clients to other types of servers (including mainframes). WRQ applied for and received the "Designed for Windows 2000 Professional" logo for each of these six programs, meaning that Microsoft has certified that the WRQ products meet Microsoft's quality assurance standards for interoperation with Windows.

79. Customers that use client software of the type just described typically can install it automatically on each Windows client computer using tools provided by Microsoft and others.[58] Microsoft also sells software for this purpose, Microsoft Services for UNIX, which includes client software to connect to servers using NFS, among other interoperability features.[59]

---

[55] Novell licenses its servers at a price determined in part by the number of clients connected. However, the licensing fee does not vary depending on whether the customer uses Novell's client software or that provided by Microsoft.

[56] "Solstice Network Client," http://www.sun.com/software/solstice/netclient/ (downloaded April 4, 2002).

[57] "Reflection NFS Client," http://www.wrq.com/products/reflection/rnfs/Welcome.html (downloaded April 4, 2002).

[58] *See, e.g.*, "Microsoft Systems Management Server," January 29, 2002, http://www.microsoft.com/smserver/evaluation/overview/default.asp (downloaded April 2, 2002). Similar client software is also available from various vendors to let Unix, Linux, or Mac OS clients connect to Windows servers.

[59] "Services for UNIX Component Summary," November 15, 2000, http://www.microsoft.com/windows2000/sfu/sfuincluded.asp (downloaded April 2, 2002).

80. In solutions to interoperability issues that rely on added client software, the client software relies on the client operating system's APIs in the same way that applications do. The APIs documented for Windows clients provide a rich set of functions that such software can use to give access to the functionality available on another vendor's server operating system. When Sun released its Solstice client, it issued a white paper that described in detail how it had "seamlessly integrated" its Solstice software using Windows APIs, thereby enabling Windows clients to obtain services from Sun's Solaris servers. According to Sun, APIs in Windows client operating systems (1) "allow them to be linked to different types of networks," (2) enabled Sun to "integrate various network services seamlessly into Windows 95 or Windows NT," and (3) allow "multiple File System Drivers"—such as Sun's NFS file sharing and Microsoft's CIFS—"to be resident in the system simultaneously."[60]

81. Novell's client software similarly demonstrates how documented Windows APIs permit high levels of interoperability. Figure 4 shows screen shots of two different file properties dialog boxes from two client PCs running Windows 2000 Professional: one that has Novell's client software installed and is connected to a NetWare server, and one that is

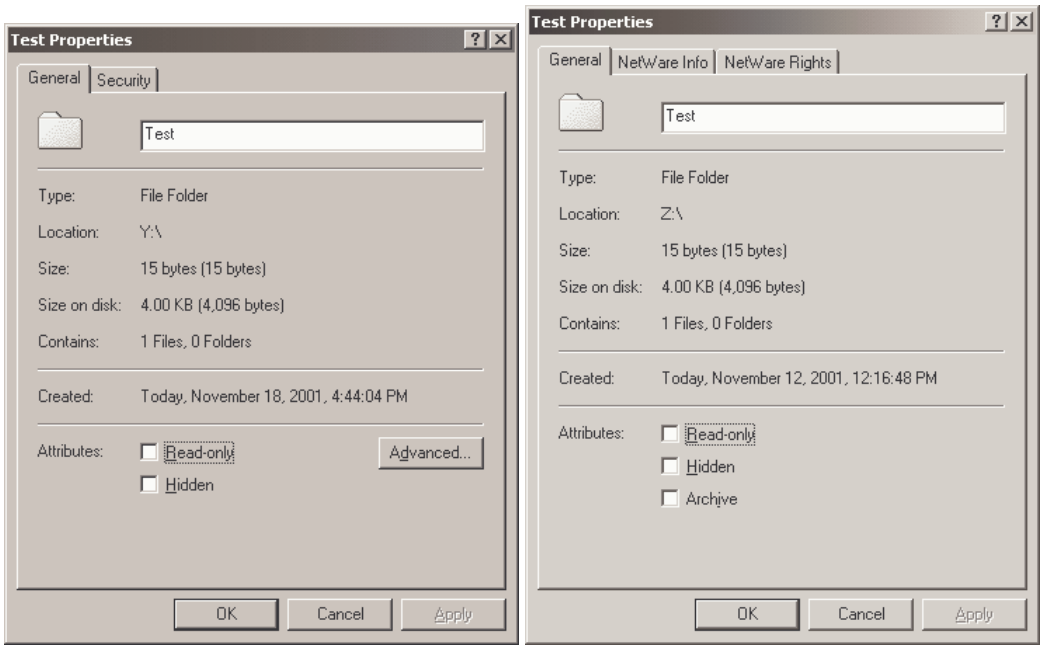
---

[60] "Solstice Network Client Technology Backgrounder," (emphasis in original), February 1998, http://www.sun.com/software/white-papers/wp-snc.tech.bg/sncpluswp.pdf (downloaded October 18, 2000):

"Windows 95 and Windows NT provide several network interfaces that allow them to be linked to different types of networks. The provider interface enables non-Microsoft network vendors to link to Win32 browsing and file I/O APIs. Solstice Network Client software is implemented as a set of full function providers. **A provider is software that establishes Windows 95 and Windows NT as a client of a remote network server**." (p. 5.)

"… [T]he MPR [Multiple Provider Router] uses the Network Provider Interface, a set of APIs used by Windows 95 or Windows NT to request network services (such as browsing services, connecting to servers, etc.). The **Network Provider Interface (NPI)** enables a Network Provider to integrate various network services seamlessly into Windows 95 or Windows NT." (p. 7.)

"… Windows 95 and Windows NT were designed with a set of APIs that are used to implement different file systems into the operating system. The Installable File System (IFS) Interface was built into Windows 95, and the File System Driver (FSD) Interface was built into Windows NT. These interfaces were designed for multiple File System Drivers to be resident in the system simultaneously. … The [Sun] FSD [File System Driver] uses the Microsoft TCP/IP stack as transport." (pp. 8-9.)

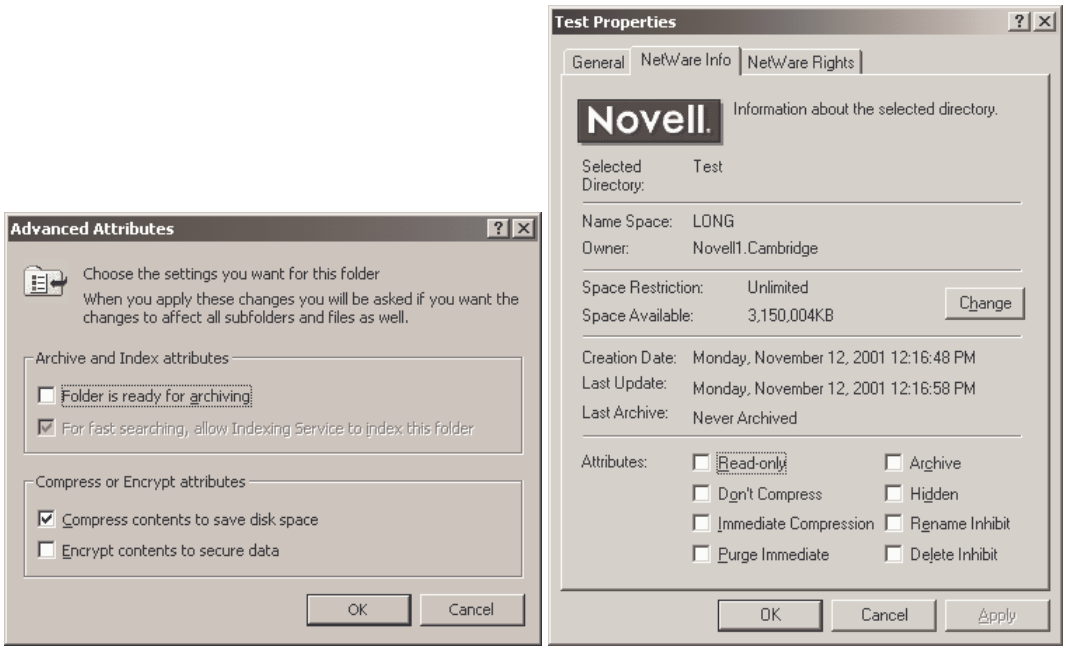


Source: Screen shots taken from Windows 2000 Professional attached to a Windows 2000 server (left) and, with Novell's client software installed, attached to a NetWare server (right).

Figure 4. Comparison of File Properties Dialog Boxes: Windows 2000 Server and NetWare Server

connected to a Windows 2000 server.[61] The picture on the left-hand side is the dialog box for a file folder on a Windows 2000 server. (It would be the same if the file were on the Windows 2000 client itself, because they use the same system.) If the Windows 2000 client were connected to a non-Windows server and did not have Novell's software installed, the dialog box would not have the "Advanced" button or the "Security" tab, because those features are specific to Windows 2000. The picture on the right-hand side of Figure 4 is for a file folder containing the same files on a NetWare server. Instead of the "Advanced" button, Novell added a tab for "NetWare Info." It also replaced Microsoft's "Security" tab with a "NetWare Rights"

[61] These screen shots were originally created for submission to the European Commission in David S. Evans, Albert L. Nichols, and Atilano Jorge Padilla, "An Economic Analysis of the Statement of Objections Filed against Microsoft" (Case No. COMP/C-3/37.792/Microsoft), November 19, 2001, ¶¶ 239–243. I have reviewed the details of how they were created with the NERA research staff who did the original work.



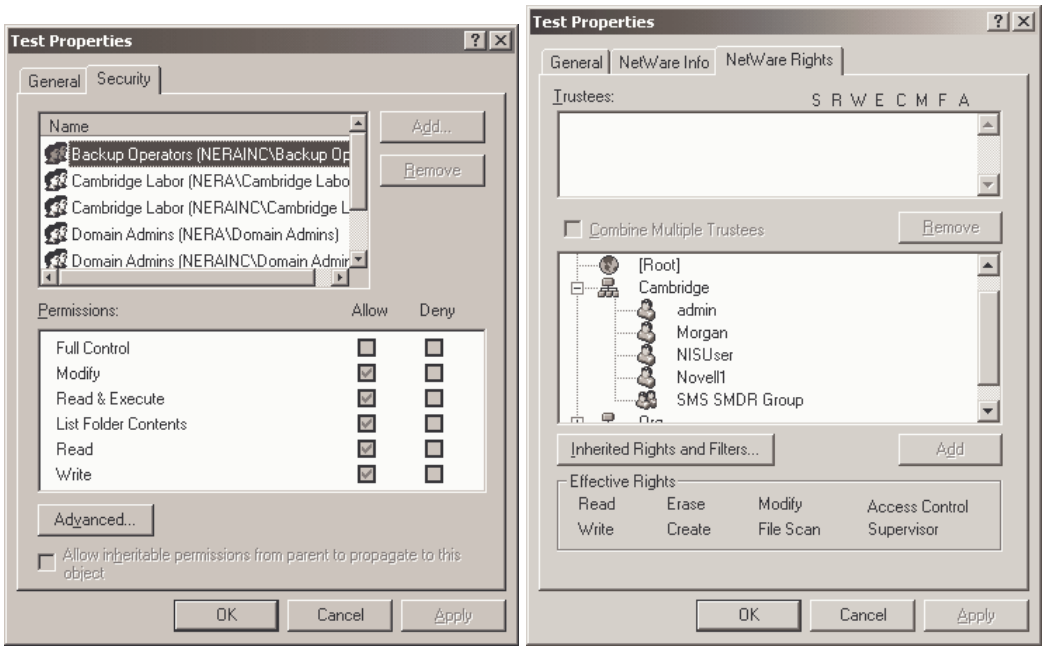
Source: Screen shots taken from Windows 2000 Professional attached to a Windows 2000 server (left) and with Novell's client software installed attached to a NetWare server (right).

Figure 5. "Advanced Attributes" and "NetWare Info" from File Properties Dialog

tab. This illustrates that Windows APIs provide considerable capabilities for competing server operating system vendors to integrate and expose their unique capabilities.

82. The left-hand side of Figure 5 shows the dialog displayed when the user clicks on the Advanced Properties button. There are check boxes for four properties: archiving, indexing, compression and encryption. Novell's equivalent tab, shown on the right side of Figure 5, includes additional information and options relevant to NetWare; it omits the Windows 2000 Server options that do not have parallels in NetWare (indexing and encryption). Note, however, that it includes more options for file compression, because NetWare handles compression differently than does Windows.[62] Besides demonstrating this well-established

[62] NetWare (on the server) generally compresses files automatically if they have not been accessed for a specified period. Thus, instead of a simple yes-no checkbox, the NetWare dialog asks if the user wants to specify that the file should never be compressed or, alternatively, wants to compress it immediately, rather than having NetWare wait to *see* if it goes unused.



Source: Screen shots taken from Windows 2000 Professional attached to a Windows 2000 server (left) and with Novell's client software installed attached to a NetWare server (right).

Figure 6. Comparison of Security/NetWare Rights Tabs in Windows 2000 Professional File Properties Dialog Box

approach to achieving interoperability, Figure 5 also illustrates that different server operating systems rarely provide identical services—the uniqueness of each system's functionality is one of the ways that vendors compete for customers. Under the non-settling States' Proposal, of course, every vendor *other than Microsoft* could still differentiate its products in this way. *Only Microsoft* would have to share its distinguishing features or innovations with its competitors.

83. Figure 6 compares the Windows 2000 security tab to Novell's corresponding "NetWare Rights" tab. The layouts are very different, because the two vendors have designed different security systems reflecting their views of what users will find most valuable. That is, *Novell competes by offering users an alternative, rather than by mimicking Windows.*

84. Note that if Novell did not include software to be installed on the client, there would be no way for the user to gain access to Novell's distinctive additional features. Novell spells out this advantage in a technical white paper on NetWare 6's file protocols:

> Take a look at the list of NetWare file attributes, then compare those to the file attributes in Windows, Macintosh, UNIX, or Linux networks. Who has more control over files? NetWare, by a large margin. You will lose that control if you don't have Novell client software in the loop.[63]

Adding software to the client gives Novell much greater flexibility in the innovations it introduces. Even if it wants to match a particular feature in Microsoft's server operating system (*e.g.*, if it wanted to add encryption), it need not duplicate Microsoft's method of implementation; it can implement the feature in whatever fashion it thinks best and have its client software use Windows APIs to add controls to the Windows client's dialog.[64]

85. Note that Novell has added these features to the Windows client using documented APIs. It did not require access to the source code or a license to obtain additional "Technical Information" on internal elements of Windows. It also is important to recognize that even if Microsoft wanted to "hide" such points of attachment from direct competitors such as Novell, it could not do so without also hiding them from all application developers. And that would diminish the value of its platform.

E. "Interoperability" Is Not Perfect Interchangeability.

86. As I discuss in the next section, many of the complaints made against Microsoft concerning "interoperability" implicitly equate that term with "interchangeability." The non-settling States' Proposal adopts this extreme approach, defining "Interoperate" as "the ability of two products to effectively access, utilize and/or support the full features and functionality of one another."[65] Under this definition, which Microsoft's competitors have championed in this

---

[63] "Novell NetWare 6: File Protocol Support Section," http://www.novell.com/info/collateral/docs/4621202.01/4621202.pdf (downloaded November 6, 2001).

[64] Microsoft's operating systems decrypt files before transferring them from one computer to another, even when both are running Windows. As a result, there is no need for the two computers to use the same encryption system. (*See* "Encrypting File System Overview," http://www.microsoft.com/TechNet/prodtechnol/winxppro/proddocs/encrypt_overview.asp?frame=true (downloaded April 8, 2002).)

[65] Plaintiff Litigating States' First Amended Proposed Final Judgment, March 4, 2002, § 22.q.

proceeding and in other forums,[66] Windows clients are deemed to be unable to "interoperate" with non-Microsoft servers unless Microsoft's competitors have the information needed to ensure that their operating systems can provide Windows clients with every service that a Windows server provides. Moreover, the non-Microsoft server must be able to mimic or clone Microsoft's technologies so closely that the Windows client "thinks" it is connected to a Windows server. Thus, the goal is to be able to provide "drop-in" or "plug-in" replacements for Microsoft's servers in the guise of the laudable-*sounding* goal of "full bilateral interoperability."

87. For example, the non-settling States' definition requires that for a Windows client running on Intel hardware to be "interoperable" with a Sun server, running a different operating system (Solaris, Sun's variant of Unix) on different hardware (Sun's SPARC microprocessor), the Sun server should be able to replace an Intel-based server running Windows in a network without having to make any modifications to software running on any other server or client computer. As I discuss in more detail below, the non-settling States' Proposal goes even further, effectively requiring that competitors have the information needed to offer perfect replacements for arbitrarily small elements of the Windows operating system.

1. Perfect Interchangeability Does Not Exist.

88. "Replaceability" or "interchangeability" goes well beyond what IT professionals normally mean when they speak of interoperability. This kind of interchangeability is not achieved even across servers produced by companies that use similar hardware and system software. For example, many companies produce servers that use RISC (Reduced Instruction

[66] *See, e.g.*, Sun Microsystems, "Application for the Initiation of Proceedings against Microsoft Corporation pursuant to Article 3 of Regulation 17/62 to Establish the Existence of Infringements of Article 86 of the Treaty of Rome," December 10, 1998, ¶¶ 9, 11 and § VII; Novell, "EU Questionnaire, Novell Inc., Case No. COMP/C-3/37.792—PO/Microsoft (Windows 2000)," July 28, 2000 (particularly the answer to question 7a).

Set Chip) processors and variants of the same basic operating system (Unix). Despite these commonalities, however, the servers these companies produce are different enough that most applications running on one must be modified to run on another vendor's system. As a result, most Unix applications are offered in versions for only a few variants of Unix, and customers choose among Unix vendors in part on the basis of the specific applications available for that platform.[67]

89. As a result, companies do not regard Unix servers as interchangeable ("interoperable" in the extreme sense the term is used by the non-settling States) and do not casually mix Unix servers from different vendors, although they may have a mix due to specialized requirements or historical reasons. Amazon.com, for example, used Sun Solaris servers for its online catalog and order processing, but then switched to Hewlett-Packard HP-UX servers in May 2000.[68] It has since switched to Intel-based servers from HP running Linux for its Web serving, while keeping its HP-UX servers for core services.[69] It did not choose to mix two brands of Unix servers for the single purpose of providing Web serving or core services. Where all of the computers must work closely together on a common set of tasks, tight-coupling is essential and there are clear advantages to uniformity. The coupling required between core services and Web serving is less tight, making it possible to mix Linux and HP-UX—though even there Amazon chose to buy both types of servers from the same vendor.

---

[67] According to Booz-Allen & Hamilton, "enterprise application availability" is significant on HP-UX and Solaris, limited on Non-Stop UNIX, SCO UNIXWare and IRIX, and somewhere in between on AIX and Tru64 UNIX. (*See* Barry Jaruzelski, Gerald Horkan and Randy Lake, "Linux: Fad or Future?" Booz-Allen & Hamilton, p. 7, http://extfile.bah.com/livelink/livelink/61801/?func=doc.Fetch&nodeid=6180 (downloaded November 12, 2001).)

[68] "HP outbids Sun for Amazon business," *Reuters*, May 31, 2000, available at http://www.zdnet.com/filters/printerfriendly/0,6061,2579571-92,00.html (downloaded October 25, 2001).

[69] Stephen Shankland, Robert Lemos, and Margaret Kane, "Amazon: Linux saved us millions," *ZDNet News*, October 30, 2001, available at http://www5.zdnet.com/zdnn/stories/news/0,4586,5098989,00.html?chkpt=zdhpnews01 (downloaded October 31, 2001).

90. Sun has been a major proponent of having the government or the courts force Microsoft to provide the information it needs to develop drop-in replacements for Microsoft's server operating systems.[70] In its marketing, however, Sun has recognized that such perfect interchangeability is not achieved, even when the heterogeneous systems come from a single vendor. In a full-page advertisement that appeared in *The Wall Street Journal* and other publications in the fall of 2001, Sun emphasized the benefits of a homogeneous (Sun) environment. Under the headline, "Do you have the Big Blues over your IT infrastructure?" Sun chided IBM for having "fourteen operating systems, multiple chip architectures and a tangle of middleware to deal with." In contrast, "Sun's systems run on one chip architecture and a single operating environment, so you can scale from under-$1,000 desktop systems to over-$10-million data center systems without breaking a sweat."[71]

2. There Are Many Reasons for Lack of "Full Interoperability."

91. Limitations on perfect interoperability among different systems reflect a host of technical and business factors on both the vendor- and customer-side of the equation.

a. Some Technical Factors

92. On a technical level, there are many different ways to accomplish any given task. If two teams of engineers/developers are given the same complex problem to solve independently, the probability that they will come up with identical solutions, with modules that can be freely interchanged, is nil. Earlier, I discussed how different operating systems use different file systems. They also manage memory differently and expose different APIs. Parallel differences exist with respect to database managers, another type of software platform.

---

[70] *See, e.g.*, Sun Microsystems, "Application for the Initiation of Proceedings against Microsoft Corporation pursuant to Article 3 of Regulation 17/62 to Establish the Existence of Infringements of Article 86 of the Treaty of Rome," December 10, 1998, ¶¶ 9, 11, 145.

[71] Sun Microsystems advertisement, *The Wall Street Journal*, October 4, 2001, p. B5.

Oracle, IBM and Microsoft, for example, all produce enterprise databases that can run on Windows servers. Oracle and IBM also offer their database software in versions for other operating systems, including variants of Unix. It is relatively easy to write modules using Oracle's 9i database software running on one machine tightly coupled to those running on other machines with very different operating systems—say Windows 2000 and Solaris or Linux. The same is true for modules written in IBM's DB2. But no one expects tight coupling (at least not without a great deal of work and expense) between Oracle databases and IBM databases, even if they are running on the same machine.

b. Some Business Factors

93. Vendors of software have strong business incentives to differentiate their products and to offer new features that appeal to customers and generate sales. Those incentives drive innovation, leading to more capable products that deliver more services at lower costs. Few, if any, server vendors want to compete solely on price, with homogeneous feature sets and complete interchangeability. Such a business model is unlikely to be profitable, since it leads to intense downward pressure on prices. It also deprives consumers of the benefits of innovation that result from competition on features and capabilities, rather than price alone.

94. Differences in customers' preferences also contribute to less than perfect interchangeability. Organizations incur the cost and effort of maintaining heterogeneous networks for various reasons. One is simply history—they have large investments in older server systems and do not want to pay to replace all of those systems and their associated application software, much of which may be custom-written.[72] Another reason, however, is that

[72] Lisa Vaas, "Keeping Air Force Flying High," *eWeek*, October 22, 2001, available at http://www.eweek.com/print_article/0,3668,a%253D16944,00.asp (downloaded October 29, 2001) ("Rewriting the systems from scratch would have eaten up an impermissibly large chunk of the Air Force's budget. 'We don't have the money to go out and say, "OK, let's wholesale replace everything,"' Jones said.").

systems have varying strengths and weaknesses, and the relative importance of those characteristics depends on the particular use of the systems. For example, the availability of inexpensive, easy-to-use commercial application software is likely to be a key consideration in choosing client systems used by non-technical employees. But the availability of prepackaged applications is much less important for servers running enterprise applications, many of which are custom-designed and managed by IT professionals and are rarely encountered directly by unsophisticated end users.

3. Countervailing Forces

95. Despite these technical limits on interoperability and business incentives for differentiation, improved technologies have made greater interoperability possible and the market increasingly imposes limits on the extent to which vendors can differentiate their products in incompatible ways. In the old days, customers accepted that IBM and Burroughs mainframes, for example, could not interoperate in meaningful ways. Today, however, in addition to innovation (and lower prices), customers prize the capacity of computers to interoperate with products from other vendors (including older versions of the vendor's own hardware or software). This demand arises not only because customer are loath to replace old computing resources, but also because many organizations need to be able to tie together computers with different characteristics and strengths.

96. Thus, along with commercial and technical pressures to differentiate products, there are also strong pressures to make new products interoperable with existing products from other vendors. As a result, many vendors offer new products that include interoperability tools. Vendors are especially eager to help customers make the transition from other vendors' products or from older versions of their own products. Sun, for example, has announced its Mainframe Affinity Program to help customers who want to migrate from IBM mainframes to

Sun Solaris servers.[73] As discussed earlier, Microsoft has for many years included software with its client operating systems that lets those operating systems connect to Novell NetWare servers, and it offers add-on products to promote interoperability between its own clients and servers and various Unix servers.[74]

97. I am not aware, however, of any vendor that offers interoperability features or products with the goal of making it possible to buy a competitor's product without losing some of the new features incorporated in its own offerings. Sun, for example, does not provide detailed "Technical Information" (such as reference implementations) on the internal design of its iPlanet Directory Server software, which protects the intellectual property it has in that product's features. Not surprisingly, Sun has little interest in allowing customers gain all of the functionality of iPlanet Directory Server without buying the product or obtaining it bundled with a Sun server system.

4. Microsoft's Operating Systems Are Becoming More Interoperable, Not Less.

98. Since Microsoft released Windows 2000 two years ago, several competitors have claimed that the new operating system "degrades" interoperability with their servers, both in terms of client-server and server-server interactions. Many of those claims have been repeated in this case, both in submissions to the non-settling States that Microsoft obtained through discovery and in testimony offered in court by competitors. For example, Dr. Ledbetter of Novell complained that interoperability with Microsoft operating systems has gotten more

[73] "…Sun Microsystems, Inc. today announced significant enhancements to its Mainframe Affinity Program, acquiring the mainframe rehosting business of Critical Path, Inc. Sun customers can now quickly rehost IBM mainframe applications onto an open, scalable platform; there are more than 500 customers currently using this software." (*See* Sun Microsystems Press Release, "Sun Introduces World's Best Solution for Attacking Cost of Ownership in Corporate Data Centers," September 25, 2001, http://www.sun.com/smi/Press/sunflash/2001-09/sunflash.20010925.1.html (downloaded October 25, 2001).)

[74] *See* III.D.2.d.

difficult with recent versions.[75] As I discuss below, however, those complaints are fairly transparent demands that Microsoft be forced to divulge intellectual property so that competitors can more easily duplicate attractive and innovative technologies that Microsoft developed at its own risk and expense.

99. Competitors (including several who have testified, as I discuss below) say that Microsoft degraded interoperability in Windows 2000, and link many of the alleged problems to two new features in that product—Active Directory and the Kerberos security protocol—with Microsoft's innovative extension for authorization. But results from the recent survey of senior corporate IT professionals by Mercer Management that I referred to earlier contradict this claim. Mercer asked whether Windows 2000 Professional (the client version) made interoperability harder or easier compared to earlier versions. Respondents chose "easier" by a seven to one margin.[76] When asked the same question about Windows 2000 Server, they chose "easier" by a margin of almost five to one.[77] Respondents were also asked why they thought Windows 2000 made it easier or harder to achieve interoperability. Almost six times as many respondents thought Active Directory made interoperability easier as thought it made interoperability harder. Fewer respondents singled out Kerberos as a factor in their judgments about interoperability, but among those who did five times as many mentioned it favorably as unfavorably.[78] These findings are consistent with my own view that successive versions of Windows have made interoperability increasingly *easier*, *not harder*.

---

[75] Trial Transcript, March 28, 2002, a.m. session (Ledbetter Redirect), pp. 1759-1760 ("It tends to get harder to interoperate with Microsoft over the period of products where I have been pretty closely involved from Windows 98 to Windows 2000 to Windows XP.").

[76] Mercer Management Consulting, "Managing Interoperability in Heterogeneous IT Environments: Primary Customer Research Findings," December 2001, p. 36.

[77] *See id.*

[78] *See id., p.* 39.

100. As I discuss in more detail below, Active Directory enables the exchange of information among directories using industry-standard APIs, and Microsoft's addition of Kerberos to its operating systems also eases interoperability in networks that use Kerberos for security. It appears that IT professionals do not regard Microsoft's new features as hindering interoperability simply because competitors have not yet figured out how to match all those features. In the Mercer survey mentioned earlier, IT professionals who had some combination of Microsoft, Novell, and Sun servers rated the three vendors roughly equally in terms of facilitating interoperability with the others.[79] In client-server interoperability, not surprisingly, respondents report the highest levels for Windows clients connected to Windows servers, but only 8 percent gave low ratings (1 or 2 on a 5-point scale) for interoperability between Windows clients and NetWare or Unix servers. Moreover, when rating interoperability between Unix servers (such as Sun Solaris or IBM AIX) and various types of client operating systems, the rating for Windows clients was slightly better than for non-Windows clients.[80]

IV. Microsoft's Competitors Seek the Ability To Replicate and Replace Windows Features, Not Interoperate with Windows.

101. Microsoft's competitors have claimed that Microsoft fails to provide the information they need to make their products interoperate with Microsoft's operating systems and "Middleware." I have reviewed the technical aspects of several of these claims in some detail, focusing on those related to interoperability in computer networks. As I explain below,

[79] *See id., p.* 35. Mercer has respondents who had the relevant combinations to rate the vendors. In all three cases, the ratings were very similar, with no consistent pattern. Respondents with Sun-Microsoft combinations rated Sun slightly better on interoperability, but those with Microsoft-Novell combinations rated Microsoft above Novell, and those with Novell-Sun combinations rated Novell ahead of Sun. If the three ratings had been done by the same respondents, these results would imply a logical inconsistency. Because the ratings were done by different groups of respondents, however, they simply show that the differences were too small to mean anything.

[80] *See id.* p. 33.

these complaints result from equating "interoperability" with "interchangeability". None of the witnesses who testified for the non-settling States is a customer (e.g., a corporation with interoperability issues in its network) or an ISV that writes applications to run on Windows. With the exception of one OEM (Gateway), all are Microsoft's competitors in operating systems (*e.g.*, Sun, Red Hat, and Novell), services (*e.g.*, AOL and SBC), or "Middleware" (*e.g.*, RealNetworks). And their complaints about "interoperability" for the most part do not concern the ability of their software to run well on Windows, but rather about lack of access to information that would facilitate the duplication of functionality provided by Microsoft's products.

A. Kerberos

102. Several witnesses for the non-Settling States—including Dr. Ledbetter, Mr. Schwartz of Sun Microsystems, and Mr. Tiemann of Red Hat—have mentioned Microsoft's extensions of one aspect of the Kerberos security protocol as an example of Microsoft's unwillingness to provide information needed to achieve interoperability or of the need for requiring Microsoft to adhere to industry standards.[81] Those complaints should be addressed almost entirely by Microsoft's recent announcement that it would document its extension of Kerberos.[82] Nonetheless, it is instructive to review the complaints for what they tell us about the nature of Microsoft's behavior and the larger agenda of Microsoft's competitors.

103. Kerberos is a method for "authenticating" users (*i.e.*, confirming that a user is who he says he is) in networks. Once a user is *authenticated* using Kerberos, the authenticating

[81] Direct Testimony of Dr. Carl S. Ledbetter, March 27, 2002, ¶¶ 98-105; Direct Testimony of Jonathan Schwartz, March 28, 2002, ¶¶ 46, 116-119; Direct Testimony of Michael Tiemann, March 21, 2002, ¶¶ 158-166.

[82] Deposition of James Allchin, February 13, 2002, pp. 156-157; John Brezak, "Utilizing the Windows 2000 Authorization Data in Kerberos Tickets for Access Control to Resources," February 2002, http://msdn.microsoft.com/library/en-us/dnkerb/html/msdn_pac.asp?frame=true (downloaded April 2, 2002).

database issues the user a "ticket," which he can then use to verify his identity to other parts of the network that use Kerberos. Kerberos was developed at MIT and has become an Internet Engineering Task Force standard.[83] The published Kerberos standard, however, does not include specifications for storing *authorization* information—that is, information as to which network resources the user is allowed to employ in various ways.[84] For example, in most networks, different users have different privileges with respect to files stored in different locations. I am allowed to edit, add, or delete files in some parts of MIT's network. In others, I can read files, but not change them. And in still others, I do not have the right to see lists of the files stored or even the directory structure that shows how they are organized.

104. Traditionally, each computer resource has managed its own list of authorizations. For file access, the server's operating system typically maintains lists of users with access to files stored on different parts of that computer's storage devices.[85] In the standard Kerberos implementation, those existing methods of authorization continue to be used.

105. For purposes of authentication, Microsoft has implemented the most recent version of Kerberos (version 5) according to the published standard. No witness has complained about this. Microsoft, however, also extended Kerberos in an innovative way to include authorization information. The information is encrypted in an unused field of the Kerberos specification, a field that was included *with precisely the expectation that future implementations would use it for authorization*.[86] Sun, Novell, and Red Hat (among others)

---

[83] J. Kohl and C. Neuman, "The Kerberos Network Authentication Service (V5)," September 1, 1993, http://www.ietf.org/rfc/rfc1510.txt?number=1510 (downloaded April 2, 2002).

[84] Although it does provide an optional field (called the "authorization-data" field) in which to place the system-specific authorization data. (*See id.*, p. 47.)

[85] *See, e.g.*, Robert E. Walsh, "Single Sign-On in Windows NT, 2000 & the Solaris Operating Environment: A Review," Sun Microsystems, January 1, 2001, p. 25, http://www.sun.com/govt/army/aes/SingleSignon.pdf (downloaded October 15, 2001).

[86] J. Kohl and C. Neuman, "The Kerberos Network Authentication Service (V5,)" September 1, 1993, p. 47, http://www.ietf.org/rfc/rfc1510.txt?number=1510 (downloaded April 2, 2002) ("The authorization-data field is

(continued...)

complain that Microsoft did not disclose how it used that field, making it impossible for other operating systems to use precisely the same method for authorization.[87] There is nothing, however, to stop them from using the same field to implement their own authorization scheme, as at least one other vendor of software that uses Kerberos has done.[88]

106. Microsoft's implementation of Kerberos is interoperable with standard implementations in most important respects. Moreover, the addition of Kerberos capabilities in Windows has made interoperability between non-Microsoft servers with standard Kerberos and Windows clients easier, not harder. For example, a Windows 2000 (or XP) Professional client can use its built-in Kerberos client software to log in to a non-Microsoft Kerberos server and obtain services in the same way that a non-Windows client (running Kerberos client software) would. In contrast, earlier versions of Windows required that Kerberos client software be added before they could log into a Kerberos server. Thus, in networks using only non-Microsoft servers and standard Kerberos, Microsoft's extensions to standard Kerberos are irrelevant; there is only a gain from the fact that separate Kerberos software is not needed on the Windows client to permit it to be authenticated to a Kerberos realm.

107. Similarly, if a user uses Kerberos to log onto a Windows 2000 server, the Kerberos ticket issued is readable and usable by non-Microsoft implementations of Kerberos; the latter implementations simply ignore the field with the authorization information (which

---

(...continued)
used to pass authorization data from the principal on whose behalf a ticket was issued to the application service. If no authorization data is included, this field will be left out. The data in this field are specific to the end service. It is expected that the field will contain the names of service specific objects, and the rights to those objects. The format for this field is described in section 5.2.").

[87] Direct Testimony of Jonathan Schwartz, March 28, 2002, ¶¶ 46, 115-119; Direct Testimony of Dr. Carl S. Ledbetter, March 27, 2002, ¶¶ 8, 95-104; Direct Testimony of Michael Tiemann, March 21, 2002, ¶¶ 161-166.

[88] "Introduction to DCE," http://www-4.ibm.com/software/network/dce/library/publications/dceaix_22/a3u2s/A3U2SM24.HTM (downloaded April 9, 2002).

would be blank anyway if the ticket had been issued by a non-Microsoft Kerberos server). Those servers perform the authorization function in whatever manner they otherwise would.

108. Other approaches to using Kerberos in networks containing Windows 2000 servers are also possible. A recent report from MIT contains a nice description of how the university addressed its demanding requirements in connection with Kerberos.[89] A couple of points from that report are particularly relevant here.

109. It is worth noting that Kerberos itself has evolved over the years, and subsequent versions (including the two most recent ones) have not always been compatible with each other. These incompatibilities are the result of changes deemed necessary by the standards bodies, and have nothing to do with Microsoft's actions. Despite these differences, MIT has found ways to support both versions in its network and prefers doing so to updating all of the systems that use the previous version.

110. In deciding how to support Microsoft's implementations of Kerberos in the campus network, MIT weighed various options. MIT was able to support Microsoft's Windows 2000 version of Kerberos, although some changes were necessary to MIT-specific software. One conclusion from the report is especially relevant to the general discussion of interoperability: "The true measure of interoperability can only be performed by each customer or user of a protocol. The question to be asked is, 'does this meet my needs?'" The report goes on to say that interoperability, in general, often is not perfect, but "… by many other metrics Microsoft has provided a highly interoperable Kerberos implementation that will meet many customer needs."[90] It definitely met MIT's needs, since clients running Windows 2000

[89] Paul B. Hill, "Kerberos Interoperability Issues," LISA-NT Conference, July 30-August 2, 2000, http://www.usenix.org/events/lisa-nt2000/hill/hill.pdf (downloaded March 22, 2002).

[90] *See id.*

Professional now log on to non-Microsoft Kerberos key distribution centers (KDCs), which in turn exchange information with Windows 2000 Kerberos installations using "trust" relationships designed into Windows 2000. As a result, Windows 2000 and other client computers seamlessly obtain services from various types of servers, and Windows 2000 clients are able to obtain the additional services possible when connected to Windows 2000 servers.

111. As the MIT case illustrates, even before Microsoft released additional information on its extension of Kerberos, customers could obtain the interoperability they needed. Indeed, Microsoft's implementation of Kerberos appears to be consistent with the non-settling States' Provision 16 regarding "industry standards." Microsoft implemented the Kerberos 5 standard in a way that lets clients and servers running Microsoft's operating systems exchange *authentication* information in precisely the way specified by the standard and lets computers running those operating systems exchange *authentication* information with clients or servers running non-Microsoft operating systems and standard Kerberos. Microsoft's innovative implementation *adds* functionality (*authorization*) when both the client and the server are running Windows and *does not* either prevent other authorization schemes from being used or require the customer to use the Microsoft authorization scheme.

112. It is important to note that Microsoft's implementation of *authorization* information in Kerberos is *not* inconsistent with an industry standard, because there is no standard that covers use of the optional authorization field. It is not even inconsistent with implementations of *authorization* information in implementations of Kerberos by competing vendors such as Sun, Novell, or IBM because they do not have their own implementations. Thus, the complaint is essentially that Microsoft has innovated by extending the functionality of Kerberos and competitors wish that it had shared the details of the innovation so that they could duplicate it.

B. Active Directory and Domain Controllers

113. Networks often contain many separate "directories" that are maintained by different pieces of software for various purposes. Email server software, for example, generally includes a directory (a specialized database) containing the addresses of individuals who use that server to send and receive email. A server's operating system typically maintains a directory of the resources connected to that server and may have a directory of the applications that are available to run on the server. Consolidating these directories into one unified directory for the network (which can include information shared by multiple servers and applications) offers advantages in terms of ease of administration. For example, if an individual leaves the organization, he can be removed from the consolidated directory rather than from each individual directory. Similarly, if the individual changes office locations, one change is required rather than several.

114. Active Directory, which is part of the Windows 2000 Server operating system, is Microsoft's consolidated network directory service. Active Directory is very much like a distributed database dedicated to certain kinds of information that are of global interest throughout the network ("white pages," security information, network configuration, applications, etc.). Its major competitors include Novell's eDirectory and Sun's iPlanet Directory Server, both of which pre-date Active Directory.[91] Novell first introduced its directory (originally called NetWare Directory Services and abbreviated NDS) as part of its

[91] iPlanet Directory Server was previously known as Netscape Directory Server and was released in December 1996. (*See* Netscape Press Release, "Netscape Directory Server 1.0 Fact Sheet," December 18, 1996, http://home.netscape.com/newsref/pr/newsrelease311.html (downloaded April 9, 2002).) NDS was first released in March 1993, as part of NetWare 4. (*See* "NetWare 4.0: Novell Launches Next Generation Network Operating System," *EDGE: Work-Group Computing Report*, March 15, 1993, p. 1.) *See also,* Kelly Jackson Higgins, "Directories Take Shape," *Internet Week*, May 15, 2001, available at http://www.internetweek.com/indepth01/indepth051501.htm (downloaded April 8, 2002) ("Many companies are going with Active Directory, eDirectory or iPlanet's Directory Server from the Sun-Netscape alliance as their primary directory, while integrating other application repositories or network operating system directories into it."); Direct Testimony of Dr. Carl S. Ledbetter, March 27, 2002, ¶ 53.

NetWare server operating system, but now offers versions for other operating systems as well.[92] However, one cannot buy NetWare without eDirectory.[93] Moreover, Novell extols the benefits to customers of having the directory service tightly integrated with NetWare.[94] Similarly, as discussed before, the version of iPlanet Directory Server for Sun's Solaris operating system running on Sun's SPARC processor includes some capabilities not available with iPlanet running on other types of servers.

115. All three of these directory services can exchange information using standard Lightweight Directory Access Protocol (LDAP) specifications.[95] Thus, if part of a network uses Active Directory and another part uses eDirectory or iPlanet Directory Server, the different directories can synchronize the common information that they store. Synchronization is a frequently used form of loose coupling that allows directories (which are essentially specialized databases) to share information despite the fact that they are organized differently.

116. Sun and Novell both offer software designed specifically to interoperate with Active Directory and to allow their directories to control what is stored in Active Directory. Sun sells the iPlanet Directory Server Integration Edition, and Sun states its product can synchronize with all information stored in Active Directory. Novell sells NDS Account Management for Windows 2000, which synchronizes all of the data in Active Directory. Each product allows the customer to use the corresponding directory service (iPlanet Directory Server or eDirectory) to control Active Directory. Both products claim to take all of the data in

---

[92] "Novell eDirectory: System Requirements," http://www.novell.com/products/edirectory/sysreqs.html (downloaded April 8, 2002).

[93] Deposition of Dr. Carl S. Ledbetter, February 10, 2002, p. 58.

[94] *See, e.g.*, "Novell Netware: Quick Look," http://www.novell.com/products/netware/quicklook.html (downloaded April 8, 2002).

[95] LDAP is a protocol used for obtaining information from or placing information in LDAP-compliant directories. For more information, see Gordon Benett, "LDAP: A Next Generation Directory Protocol," *Intranet Journal*, available at http://www.intranetjournal.com/foundation/ldap.shtml (downloaded April 7, 2002).

Active Directory and to store it in the vendor's directory.[96] As Dr. Ledbetter testified under cross examination, at its most recent "BrainShow" conference, several Novell customers spoke about how they used Novell's DirXML product to control large networks with multiple directories.[97]

117. I have seen two types of complaints about interoperability with Active Directory. One is that Active Directory stores authorization information that is passed in the authorization field of Microsoft's Kerberos implementation.[98] Thus, although LDAP can be used to obtain that authorization information, other vendors' directories cannot use the information as stored in the authorization field. This is really just one specific illustration of the generic Kerberos complaint already considered. As discussed above, the Microsoft-created authorization information is not needed for reasonable levels of interoperability, and in any event should be addressed by Microsoft's publication of the information relevant to the authorization field in its Kerberos implementation.[99]

---

[96] "iPlanet Directory Server Integration Edition Data Sheet," http://www.iplanet.com/products/iplanet_directory_ie/ds_directory_ie.html (downloaded October 28, 2001); "Novell: Account Management 2.1: Detailed View," http://www.novell.com/source/printer_friendly/dt14319_en.html (downloaded October 28, 2001).

[97] Trial Transcript, March 27, 2002, p.m. session (Ledbetter Cross), pp. 1667-1674.

[98] Sun Microsystems, "Application for the Initiation of Proceedings against Microsoft Corporation pursuant to Article 3 of Regulation 17/62 to Establish the Existence of Infringements of Article 86 of the Treaty of Rome," December 10, 1998, ¶¶ 61-65, 145-147. For more on Microsoft's use of the authorization field in Kerberos, see § IV.A of this document.

[99] Mr. Tiemann's testimony (¶ 152) also complains about Microsoft's ADSI (Active Directory Service Interface) software, which is installed on the client and provides a streamlined way to communicate with servers employing different types of directories. My understanding is that Microsoft currently offers four different ADSI "providers" that permit directories to be accessed from Windows operating systems: (1) LDAP (which is used with Active Directory and other LDAP-compliant directories); (2) the limited directory features of Windows NT 4 servers; (3) NetWare NDS/eDirectory; and (4) NetWare 3 Bindery (*See* "Active Directory Service Interfaces Overview," March 22, 2001, http://www.microsoft.com/windows2000/techinfo/howitworks/activedirectory/adsilinks.asp (downloaded April 3, 2002).). In addition, Microsoft publishes specifications so that developers can write their own ADSI "providers" for other directories. I am not clear as to the nature of Mr. Tiemann's complaint, unless it deals with the Kerberos issue. Mr. Tiemann seemed unable to clarify the issues on cross-examination or in his redirect examination (*See* Trial Transcript, March 25, 2002, a.m. session (Tiemann Redirect), pp. 1019-1020; Trial Transcript, March 25, 2002, p.m. session (Tiemann Cross), pp. 1157-1159.)

118. A second type of complaint is that Microsoft does not publish the information needed to permit competing vendors to implement Active Directory in their server operating systems.[100] As a result, non-Microsoft servers cannot be "domain controllers" in the portion of a network operating in Windows 2000 "native" mode. (In Microsoft's terminology, domain controllers are the servers that provide log-on services and authentication.)

119. At any given time, at least one domain controller must be active in a Windows 2000 network running in native mode. Thus, in claiming that they want "full bilateral interoperability" (to use Dr. Ledbetter's phrase) with Microsoft's newest products, Microsoft's competitors essentially are demanding that customers be able to obtain all of the functions of Microsoft's newest server operating system without having that operating system installed on any servers in the network. It is as if Microsoft demanded enough details on iPlanet Directory Server so that a Windows server could provide precisely those same services (in precisely the same way) without the customer having to license any copies of the iPlanet product. I have never seen "interoperability" used this way in a commercial context, and it is not consistent with the IEEE definition of "interoperability" with which I introduced this discussion.

120. If one rejects this extreme demand for interchangeability and replaceability, there are many workable solutions short of installing additional software on the Windows clients:

[100] Sun Microsystems, "Application for the Initiation of Proceedings against Microsoft Corporation pursuant to Article 3 of Regulation 17/62 to Establish the Existence of Infringements of Article 86 of the Treaty of Rome," December 10, 1998, ¶¶ 75, 145-147. Indeed, section 4 of the non-settling States' Proposal is designed to provide this information to Microsoft's competitors (*See* Plaintiff Litigating States' First Amended Proposed Final Judgment, March 4, 2002, § 4).

- The Windows 2000 server(s) can be run in a backward compatibility mode, in which case the Windows NT 4 conventions are used and non-Microsoft servers running Samba, Sun's PC NetLink, or similar software can serve as domain controllers.[101]
- The network can run in Windows 2000 "native/mixed mode," with Active Directory and Microsoft's Kerberos implementation running on one or more Windows 2000 servers. In a network thus configured, all of the other servers in the network can use non-Windows operating systems and provide traditional print and file services in the same way that Windows NT 4.0 servers do.[102]
- The network can use Windows 2000 in native mode for some subsets of the network, but use an alternative directory service to "control" the network as a whole. For example, Sun's iPlanet subsidiary and Novell both claim that their directory services can be used as the master directory. Changes made in those services then can be transmitted to Active Directory using documented interfaces.[103]

C. Outlook-Exchange and Other Client-Server Applications

121. Some observers, including Mr. McGeady and Mr. Tiemann,[104] complain that Microsoft hinders interoperability by not documenting all of the ways in which some of its client application software (distributed separately from the client desktop operating system) communicates with some of its server application software (distributed separately from the server operating system). Most of their attention has focused on certain types of information communicated between the personal information manager, Outlook, included in Microsoft Office, and Microsoft's email server application software, Exchange. Although Microsoft has

[101] *See*, *e.g.*, "Solaris PC Netlink Software FAQs," p. 12, http://www.sun.com/software/solutions/interoperability/netlink/faqs.html (downloaded October 17, 2001).

[102] *See*, *e.g.*, "Security = Domain in Samba 2.x,", http://us1.samba.org/samba/ftp/samba-2.2.2.tar.gz (downloaded October 17, 2001) (documentation to the source code package for Samba 2.2.2).

[103] *See*, *e.g.*, Trial Transcript, March 27, 2002, p.m. session (Ledbetter Cross), pp. 1667-1674. The reverse is also true: Active Directory can be used as the master, synchronizing other directories on other servers that control subsets of the network.

[104] Direct Testimony of Steven McGeady, March 26, 2002, ¶¶ 44-45; Direct Testimony of Michael Tiemann, March 21, 2002, ¶¶ 147-149.

documented most of the relevant interface between these two non-operating system products (an interface called "MAPI"), apparently it has not documented how Exchange uses individual calendar information to find times when a group could meet. Novell, which offers a competing client-server product (GroupWise), argues that Microsoft should disclose such information so that GroupWise servers could provide the same service to Microsoft Outlook clients.[105]

122. This situation is in many ways analogous to the issues concerning Active Directory, with the difference that the client side software is part of an application; it is not even part of the client operating system. As with Active Directory, the operative functionality resides on the server, although as on the client side, it is provided by a separate product, Exchange, not the server operating system. Exchange uses the individuals' calendars, which are stored on the server, not the client, to coordinate schedules. This is an example of tight coupling between client and server software designed to work together permitting additional functionality. The three major vendors of enterprise email services—Microsoft, Novell, and Lotus (with its Notes client and Domino server application software)—all offer such combinations of server and client software designed to work best together. And like Microsoft with Outlook-Exchange, Lotus and Novell do not disclose all of the details of how their client software works with their server software.[106] All three systems also have the ability to connect heterogeneous client-server combinations for most functions. Microsoft Outlook can send and receive email from non-Exchange servers, as can the other clients. Similarly, Exchange can deliver and receive email from non-Outlook clients, as can the other email servers. Many organizations, including MIT, use other types of email servers with a wide mix of clients,

[105] Mr. Mace, of Palm, testified, however, that Palm's handheld devices provide a calendaring system that synchronizes with Outlook, and that Palm has not experienced any significant interoperability problems relating to Outlook. *See* Trial Transcript, March 28, 2002, p.m. session (Mace Cross), pp. 1885, 1922.

[106] Trial Transcript, March 27, 2002, a.m. session (Ledbetter Cross), p. 1601.

sacrificing some of the functionality of tightly coupled client-server pairings in favor of more flexibility. For example, I use Eudora email client software running on Windows and generally connect to custom email server software running on a Unix-based MIT server.

123. Thus, the "complaint" about Microsoft is little more than a statement that Microsoft has engaged in a form of competition common to the major email software competitors. In this regard, it is worth noting again that this issue does not concern operating system interoperability at all. Moreover, the Microsoft email server application, Microsoft Exchange, does not have a particularly large share of email software. As of the end of 2000, IDC estimated that there were 291 million worldwide corporate email users, of which 42 percent used one of the simpler email servers that do not have collaboration features. The 58 percent using collaborative systems was divided among Lotus Domino/Notes (23 percent), Microsoft Exchange (20 percent), and others (including Novell's GroupWise).[107]

V. The Non-Settling States' Disclosure Requirements Promote Cloning, Not Interoperability.

124. The non-settling States' Proposal would require Microsoft to disclose extensive information to a wide range of parties, including direct competitors. In two of the provisions (Provision 12, concerning Web browsing software, and Provision 14, concerning Office), the forced sharing of intellectual property is quite explicit. However, Provision 4, "Mandatory Disclosure to Ensure Interoperability," would also result in the disclosure of far more of Microsoft's intellectual property than is required to achieve interoperability as the term is commonly used by IT professionals. (I emphasize again here my understanding that interoperability issues did not figure in any of the acts found illegal by the Court of Appeals.)

[107] IDC, *Integrated Collaborative Environments Market Forecast and Analysis, 2001-2005*, Report #24775, June 2001, Tables 3, 4, 5; IDC, *Email Usage Forecast and Analysis, 2001-2005*, Report #25335, August 2001, Tables 11, 13.

Armed with such information, Microsoft's competitors would be able to clone much of the functionality in Microsoft's products. Moreover, under Provision 15, they would be entitled to most of the information on a cost-free basis.

A. What is "Middleware"?

125. Provision 4(a) of the non-settling States' Proposal requires that Microsoft publish "all APIs, Technical Information and Communications Interfaces that Microsoft employs to enable:

> "i. each Microsoft application to Interoperate with Microsoft Platform Software installed on the same Personal Computer;
>
> "ii. each Microsoft Middleware Product to Interoperate with Microsoft Platform Software installed on the same Personal Computer; and
>
> "iii. each Microsoft software program installed on one computer (including without limitation Personal Computers, servers, Handheld Computing Devices and set-top boxes) to Interoperate with Microsoft Platform Software installed on another computer (including without limitation Personal Computers, servers, Handheld Computing Devices and settop boxes)."

To understand the implications of these requirements, it is essential first to understand how the non-settling States' Proposal defines "Middleware."

1. Important Concept: Modules as Interdependent Building Blocks of Operating Systems

126. It is important to note that although terms such as "operating system," "middleware," and "APIs" are used extensively in these proceedings, they do not have completely precise and unambiguous definitions among computer scientists. Furthermore, to the extent that there are generally accepted notions for some of these terms, they may vary from the usage in these proceedings. Figure 7 presents a highly simplified representation of an operating system as a wall built of irregularly shaped blocks. Complex software is not arranged in neat "blocks" of discrete functions, and I do not mean to suggest that this "wall" reflects the actual architecture of Windows. For example, Windows XP is built from close to 200,000

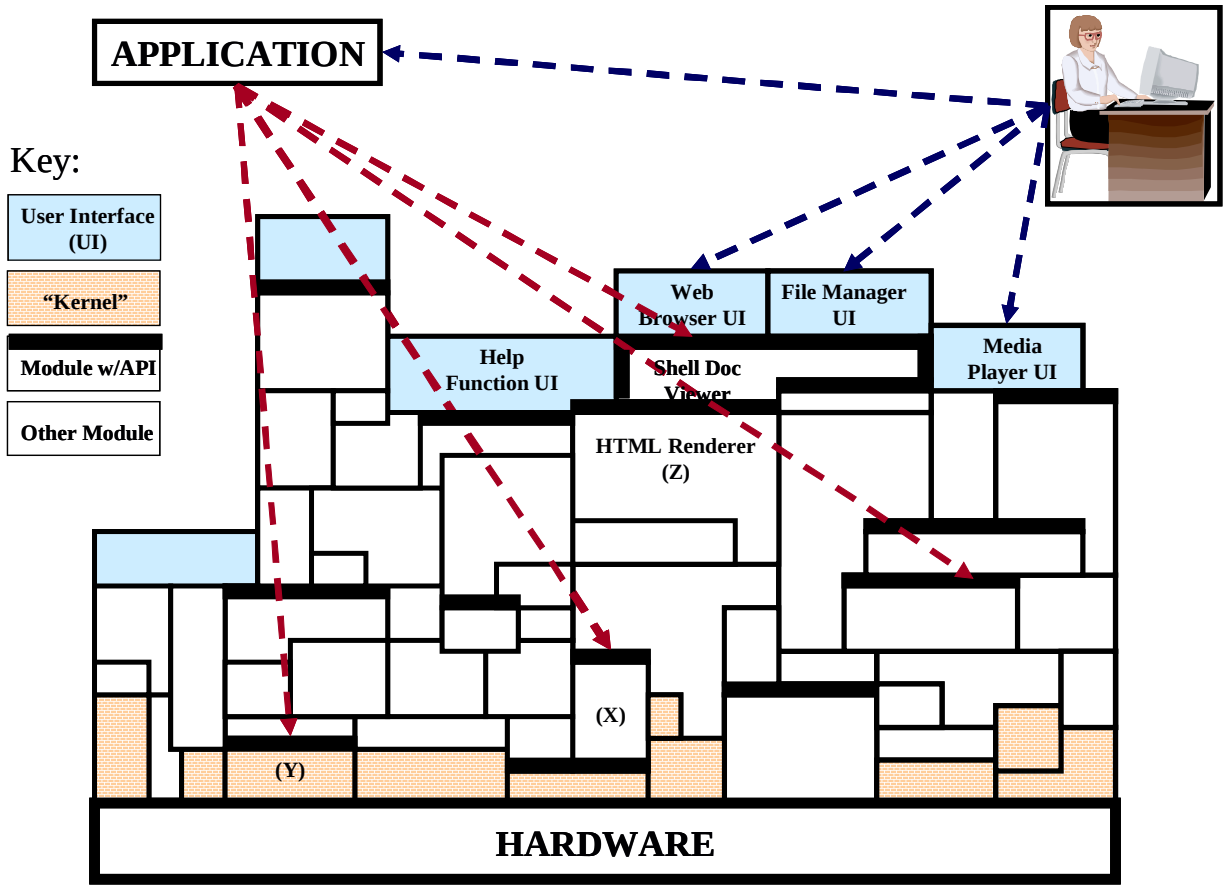


Figure 7. Operating Systems Are Built from Interdependent Modules

individual source files that are compiled into thousands of files distributed as the product, is far more complex than Figure 7 indicates, and cannot be represented completely in a two-dimensional diagram. Nonetheless, the diagram provides a useful visual heuristic, and I will refer to Figure 7 to be clear in my usage of these terms and to help explain the reasons for what might appear to be contradictory views from other documents.

127. The key concept illustrated in Figure 7 is that generally all software is developed by creating software modules (also called procedures or subroutines), shown as irregularly shaped "blocks" in the wall. Each "block" is a collection of computer instructions that performs a particular function. Most software programs, including operating systems, consist of many of these software modules. Furthermore, each module may make use of other modules, illustrated

by blocks that sit on top of other blocks to construct a wall. Unlike most walls, however, the modules in an operating system often have reciprocal dependencies, with a "block" depending not only on those "below" it, but also on some "above" it or to its "side."

128. The wall is shown consisting of a variety of blocks of different sizes and shapes to illustrate that there are many different ways to construct such a wall, and there may be no simple pattern to their arrangement. Each "block" is custom built and evolves over time as the structure is enlarged or otherwise modified. Even starting with similar overall goals, different designers of a complex software system, such as an operating system, will almost certainly come up with a different selection of blocks and ways to order and connect them to construct the system (or "wall").

129. A modern operating system such as Windows 2000 may have 100,000 modules or more. These modules may be stacked up tens or hundreds of layers deep. Removing any particular module or combinations of modules, such as X, Y, and/or Z in Figure 7, can have consequences for many other modules in the structure that depend upon them—possibly causing a collapse of the "wall." Some modules are used by many others. For example, the file shdocvw.dll (called "shell doc viewer") is called by many different interfaces, including the file manager (Windows Explorer),[108] the Web browser (IE),[109] Windows Media Player,[110] and the HTML Help System.[111] In the general case, and especially in operating systems, each module may depend upon very many modules, not just the ones shown immediately adjacent to it, creating more of a fragile lattice work—much more complex than the picture in Figure 7.

---

[108] Direct Testimony of James Allchin (liability trial), January 27, 1999, ¶¶ 100, 103.

[109] "About the Browser," http://msdn.microsoft.com/workshop/browser/overview/overview.asp?frame=true (downloaded April 17, 2002).

[110] Deposition of Linda Wolfe Averett, March 5, 2002, pp. 110-116.

[111] "HTML Help 1.3 SDK: HTML Help API Overview," http://msdn.microsoft.com/library/en-us/htmlhelp/html/vsconovhtmlhelpapioverview.asp?frame=true (downloaded April 17, 2002).

130. Each software module has one or more interfaces that enable other modules to use it. In an operating system, some of these interfaces are exposed and documented to enable application developers to use that module. These APIs are shown as the heavy lines along the edges of some blocks. The top row of shaded blocks represents the user interfaces (UI) that let people access the functionality, but it is important to note that these blocks usually are supported by and depend upon hundreds or thousands of other blocks. In many cases, these top-level modules actually may be very small and contain minimal functionality, in which case they are often called "stubs."

a. Dr. Appel's Example

131. Similar principles were introduced by Dr. Appel, the non-settling States' technical expert, in his direct testimony, and especially his Exhibit E (PX 1721). (For the convenience of the Court, I include copies of Professor Appel's exhibits B and E (PX 1718 and 1721), in Attachment C. He illustrated a module, named *client.c*, that displays all the prime numbers from 2 to 99—which we might consider to be "Middleware" if it were distributed with the operating system. The top-level module *client.c* relies upon another module, named *testprime.c*, which in turn relies upon a module named *divide*s.*c*. There are many other ways that this simple application could have been designed. For example, the functionally of *testprime.c* and *divides.c* could have been combined into a single module or split across multiple modules. Such differences would lead to different shaped blocks with different APIs in our wall analogy, and generally the individual modules would not be interchangeable.

132. It should also be noted that in order to actually display the results, Dr. Appel's top-level module uses the *printf* function, which Dr. Appel references but does not elaborate upon. This module, often provided as part of the C runtime library, is likely to call upon dozens

of other modules in the operating system to accomplish its functionality.[112] So even Dr. Appel's simple example could be quite complex.

b. What is "Middleware"?

133. In Dr. Appel's Exhibit B (PX 1718), he shows neat boxes labeled "Middleware A," "Middleware B", etc. In reality, defining a particular piece of middleware is much more complex, and will rarely be equivalent to a single identifiable file or "block" of code. For example, if we viewed his prime number program as a very simple piece of middleware, what does it actually comprise? Is it just the main module (*client.c*), or is it that module plus *testprime.c*? Do we also include *divide.c*, *printf*, and the potentially hundreds or thousands of other modules that they might rely upon? At what point do we stop?

134. These same issues, only to a vastly more complex degree, apply to the "Middleware" examples that have been listed by the non-settling States. Although it may be relatively easy to define a piece of "Middleware" (such as the "Web browser") at the "top" or user interface level, defining it in terms of the code that supports that functionality is necessarily ambiguous, without any technically determined "right" answer. When does the "Middleware" end and the "operating system" begin? What about modules that support not only the function in question, but also others—within the operating system product, in other "Middleware," and in third-party applications? This is especially challenging in a complex operating system, such as Windows (or the Mac OS), that provides many APIs that are widely used by other parts of the operating system, many of which also would be "Middleware" under the non-settling States' definition.

[112] A simple inspection of "MSVCRT.DLL"—an implementation of the C runtime library shipped by Microsoft with Windows XP—reveals that the runtime library directly uses at least 152 Windows APIs; the implementation of *printf* function itself probably directly uses only a subset of those 152 APIs. (*See* Output from running Dependency Walker on MSVCRT.DLL.)

2. The Non-Settling States' Definition of Middleware

135. My understanding is that, during the liability trial, "middleware" was used to refer to software that ran on multiple desktop PC operating systems and that exposed APIs that general purpose applications could use to run on any of those operating systems. If a piece of "middleware" provided enough APIs to support a rich set of applications, the plaintiffs argued (and the Court of Appeals agreed), it could become an alternative platform to Windows for running applications. Applications that relied on those "middleware" APIs (without making direct use of operating system APIs) could run on any PC operating system for which the "middleware" was available, making it easier for other PC operating systems to become serious competitors to Windows.[113]

136. The definition of "Middleware" in the non-settling States' Proposal, by contrast, includes *any* software that provides *any* interface through which it could provide services to other software and that "could, if ported to or made Interoperable with multiple Operating Systems, enable software products written for that Middleware to be run on multiple Operating System Products."[114] The definition lists more than a dozen examples of "Middleware," ranging from "network operating systems" to "calendaring systems."[115] These examples, however, are not exhaustive, and my reading of the non-settling States' definition indicates that any functionality—including that exposed by any one of the 6,500 or so APIs provided by Windows—could constitute "Middleware" under that definition. Thus, for example, the code that supports APIs for drawing objects on the Windows screen would appear to be "Middleware," as would the code that supports the rendering of HTML documents or the

[113] United States Court of Appeals Decision, No. 00-5212, June 28, 2001, pp. 53-54. The Court found that Java and Navigator were "middleware" threats because they could be made to or did run on multiple hardware-operating system combinations and because Java exposes many APIs (its class libraries).

[114] Plaintiff Litigating States' First Amended Proposed Final Judgment, March 4, 2002, § 22.w.

[115] *See id.*, § 22.w. It is not clear to me that all of these examples comport with the general definition given in the proposal, but they are explicitly labeled "Middleware" in the non-settling States' definition.

rendering of fonts so the page shown on screen matches the printed page. Professor Appel apparently agrees with this reading of the definition, as he testified during his deposition that it includes software exposing even a single API.[116] The person designated as "most knowledgeable" concerning the meaning of the non-settling States' Revised Proposal, Assistant Attorney General Greene of California, agreed that a module of code exposing a single API could be considered "Middleware" under the Proposal.[117] To be a useful general-purpose platform for running applications on multiple operating systems, however, "middleware" must expose a much richer set of APIs; otherwise, applications will have to depend primarily upon APIs exposed by the underlying operating system, and hence will be hard to "port" to other operating systems.

137. It is also important to note that the non-settling States' definition of "Middleware" does not require that it actually be available on multiple operating systems, only that it have the potential of being "ported" to other platforms. *But any software has that potential*. This fact, together with the "one-API" standard, means as a practical matter that virtually any feature of Windows, no matter how small, is "Middleware" under the non-settling States' Proposal.

138. Indeed, the Proposal lists only two examples of features that are *not* "Middleware": "disk compression and memory management software."[118] These examples do not illuminate the definition in any useful way, however, because it is not clear what characteristics of disk compression and memory management put them outside the non-settling

[116] Deposition of Dr. Andrew W. Appel, February 27, 2002, p. 77. During his cross examination, Professor Appel suggested that "Middleware Products" are much larger collections of code modules. *See* fn. 130 and associated text.

[117] Deposition of James Thomas Greene, March 12, 2002, p. 61.

[118] Plaintiff Litigating States' First Amended Proposed Final Judgment, March 4, 2002, § 22.w.

States' definition. Microsoft exposes APIs for applications to use these two specified features of the operating system,[119] and making those APIs available in other operating systems or "Middleware" presumably would ease the porting of applications that use those APIs.

139. Dr. Ledbetter of Novell offered perhaps a more candid—and revealing—explanation for why those two aspects of Windows were excluded. In his deposition, he suggested that a key factor is whether the feature has matured to the point that there is little interest among other firms in further development.[120] That testimony suggests the definition has been crafted to enable competitors to have access to interesting, innovative, or new features of Windows. Any application of such a "principle" would be extremely subjective. I note, for example, that one of the areas to which Professor Appel devotes his research efforts is memory management.[121] Presumably if there is a breakthrough in that field, system vendors may become more interested in memory management as a competitive edge, in which case it should have been included as "Middleware" under Dr. Ledbetter's criterion.

140. Coming at the definitional question from a somewhat different direction, one might ask: what is *not* "Middleware"? In his deposition and direct testimony, Dr. Appel equates "middleware" with everything distributed with the Operating System Product other than the "operating system kernel." He is quite explicit about this in Exhibit B (PX 1718) of his testimony. He defines the kernel as follows: "The operating system *kernel* is a software

---

[119] "Memory Management Functions," http://msdn.microsoft.com/library/en-us/memory/memman_60kz.asp?frame=true (downloaded April 9, 2002); "SetFileAttributes," http://msdn.microsoft.com/library/en-us/fileio/filesio_9a2b.asp?frame=true (downloaded April 9, 2002); "Data Decompression," http://msdn.microsoft.com/library/en-us/fileio/datadcmp_6tym.asp?frame=true (downloaded April 9, 2002).

[120] Deposition of Dr. Carl S. Ledbetter, February 10, 2002, pp. 261-263.

[121] Direct Testimony of Dr. Andrew W. Appel, March 13, 2002, ¶ 21 ("… in recent years, I have primarily focused on the software development of compilers (programs that translate source code into executable code), automated memory management (particularly the form of memory management called 'garbage collection'), and run time systems…").

component within the operating system that operates in 'privileged mode' on the computer hardware, meaning that it has access to all of the computer hardware devices so that it can mediate services to applications."[122] In Figure 7 (p. 66), this is depicted as some of the blocks at the bottom of the wall (those with a pattern of small bricks inside).

141. This definition—"the 'operating system' is the kernel"—is problematic on several levels.[123] First, the decision as to what to include or exclude from the kernel is often driven by tradeoff issues of stability versus performance. Minimizing code in the kernel generally increases stability. But each crossing of the nonprivileged/privileged line (*i.e.*, each time code outside the "kernel" calls code in the "kernel" under Dr. Appel's definition of the term) usually slows execution (because it creates a hardware "interrupt" and a certain amount of software overhead). There are, therefore, reasons to put more code in the kernel. To gain performance, one operating system designer might include a specific module as part of the kernel, whereas another designer might exclude that module from the kernel to reduce the stability risk that might occur if that module malfunctioned. Note that the functionality provided is exactly the same in both cases—the exact same modules and relationships shown in Figure 7 (p. 66) exist, only the shading of the block would change.

142. Second, as a general principle of software design, one normally tries to minimize the amount of code running in protected mode (which Professor Appel defines as the operating system kernel) to reduce the chance of "crashing" the system. So how much of any operating system product is the kernel (*i.e.*, not "Middleware") under Professor Appel's

[122] Direct Testimony of Dr. Andrew W. Appel, March 13, 2002, ¶ 27 (emphasis in original).

[123] In his cross examination, Dr. Appel appears to have modified this definition to extend beyond the kernel to some definition of a "core" operating system that also includes certain basic libraries. However, he appears to be unable to offer a clear definition of the software or functions that comprise that "core." (*See* Trial Transcript, April 9, 2002, p.m. session (Appel Cross), pp. 2991-2992; Trial Transcript, April 10, 2002, a.m. session (Appel Cross), pp. 3115-3119.)

definition? Although precise numbers are usually not reported (in part because most developers do not define "kernel" in precisely the same way as Professor Appel, or each other), I asked researchers to determine the percentage of Linux and Windows XP operating system products that were kernel—the estimates for Linux ranged from a high of 3.2 percent[124] to a low of 0.2 percent;[125] for Windows XP, the estimate was 1.5 percent.[126] In every case, it is an extremely small percentage of these operating system products. In other words, "Middleware," according to Dr. Appel's definition, is more than 97 percent of the operating system product.[127]

143. This narrow definition of "operating system," and the correspondingly broad definition of "Middleware," make little sense in terms of the "market" for operating systems that I understand is at the heart of the liability findings. There is no viable "market" for kernels.[128] I am not aware of any significant commercial operating system that distributes the kernel separately. It is also important to recognize that developers write to the full set of APIs, most of which are supported by "Middleware" libraries, not just the relatively small API set exposed by the kernel. The kernel by itself is useless to users and virtually useless to

[124] Red Hat Linux 7 comes with source code to two versions of the Linux kernel This comparison considers the larger one. Comparison of size of compressed source code package "kernel-2.2.16-22.src.rpm" (20.1 MB) to total size of all compressed packages on Red Hat Linux 7 source code CD (626 MB) minus size of compressed source code package (19.3 MB) of "kernel24-2.4.0-0.26.src.rpm". *See also*, Richard Stallman, "Linux and the GNU Project," http://www.gnu.org/gnu/linux-and-gnu.html (downloaded April 5, 2002) ("… Many users are not fully aware of the distinction between the kernel, which is Linux, and the whole system, which they also call 'Linux.' … Linux itself [the kernel] was about 3%.")

[125] Comparison of size of "/boot" subdirectory (2.7 MB) to total size of "/boot," "/bin," "/etc," "/lib", "/opt", and "/sbin" subdirectories (1303.7 MB) on a SuSE 7.0 Linux system. SuSE GmbH, *SuSE Linux 7.0: Installation, Networking, Know How* (Oakland, CA: SuSE Inc., 2000), p. 375

[126] A clean install of Windows XP Professional took up over 900 MB of disk space. (*See* Screenshot of *Windows XP: Local Disk (C:),* taken April 9, 2002). On the same system, the amount of "Kernel Memory" taken up was around 13.5 MB, for an estimate of 1.5 percent. (*See* Screenshot of *Windows Task Manager: Performance*, taken April 9, 2002.)

[127] This estimate is consistent with Dr. Appel's own estimate during cross examination, where he indicates that the kernel is "at least an order of magnitude" smaller than the operating system as a whole; *i.e.*, less than 10 percent. (*See* Trial Transcript, April 9, 2002, p.m. session (Appel Cross), p. 2971.)

[128] The Mach kernel has been widely used in various distributions of Unix, but it is not sold to users separately.

developers. No one would write applications for Windows (or Linux or Mac OS X) if those products did not include a great deal of what Professor Appel calls "Middleware."

144. I note that the newest Mac operating system, Mac OS X, is built on an open source version of Unix (the Berkeley Standard Distribution, or BSD). Apple calls this core (which includes more than what Dr. Appel would call the "kernel") "Darwin" and makes it available for free.[129] But no "Macintosh" application will run on Darwin. For those applications to run, the computer must have the rest of Mac OS X—all of which Professor Appel would call "Middleware"—to provide the user interface and most of the functionality on which applications depend. So the entire $129 that Apple charges for the client version of its Mac OS X operating system product pays for "Middleware," under Professor Appel's nomenclature. For all of these reasons, Professor Appel's definition makes no sense in either technical or business terms.

3. "Microsoft Middleware Products"

145. The disclosure requirements proposed by the non-settling States apply not to "Middleware" as a whole, but to "Microsoft Middleware Products," which might seem to limit the scope of the required disclosure. Dr. Appel in cross examination seems to use this concept in claiming that the removal (and disclosure) requirements apply to large "Middleware Product" groups of modules, and not to individual components within those groups.[130] However, "Middleware" becomes a "Microsoft Middleware Product" if it has been "distributed

[129] "UNIX Based: The Open Desktop," http://www.apple.com/macosx/technologies/darwin.html (downloaded April 10, 2002) ("At its core, Darwin uses BSD. If you're a hardcore geek, you'll like having a full command set available to you from the terminal. … Best of all, Darwin is distributed under Apple's Open Source license, so engineers around the world can help Apple make Mac OS X the best operating system on the planet." Note that when Apple distributes Darwin, it distributes not only the portion used in Mac OS X, but also a great deal of open-source BSD "middleware" that is needed to create a workable operating system that can run Unix applications.)

[130] Trial Transcript, April 9, 2002, p.m. session (Appel Cross), pp. 2980-2983.

separately from a Microsoft Operating System Product" or if it "provides functionality similar to that provided by Middleware offered by a Microsoft competitor."[131] Since virtually all Windows code provides functionality that is arguably similar to code included in at least one other operating system or other platform, there does not appear to be much distinction between Microsoft "Middleware" and "Microsoft Middleware Products."

146. Even if Microsoft were required to enable the removal only of combinations of features that correspond to existing third-party products, ultimately Microsoft would have to be prepared to remove arbitrarily small elements. For example, consider "media players," one of the "Microsoft Middleware Products" listed explicitly by the non-settling States' Proposal. I asked researchers working under my supervision to identify various media players that are available and to compare their features to those in Windows Media Player (WMP). Although the researchers did not attempt to develop an exhaustive list, they found eight media players for comparison, as shown in Table 2. Many media players cover only a subset of the 14 features listed for WMP, while others include features not in WMP. Thus, even if one ignores differences in their APIs, these media players are *not* functionally interchangeable. Depending solely on the arbitrary choice of which one served as the definitional "standard," the functionality of a "media player" would vary. This makes no sense but is the necessary implication of the definitional approach taken by the non-settling States.

147. Because of these differences, the vendors of competing media players (any of which could become a "Third-Party Licensee" by licensing 10,000 copies of Windows for sale) might well like a version of Windows that disables some WMP functions (those that compete with theirs), but not others (any that provide complementary services). If one looks at the various combinations of features that such vendors might want deleted, it quickly becomes

[131] Plaintiff Litigating States' First Amended Proposed Final Judgment, March 4, 2002, § 22.x(ii).

Table 2. Comparison of Media Players' Features

Function	WMP XP	RealOne	RealPlayer 8	RealJukebox	Spinner 4.0	MusicMatch Jukebox 7	Winamp	QuickTime Player	iTunes (Mac)
Plays Audio CDs	■	■		■		■	■		■
"Burns" to CDs									
Audio CDs	■	■		■		■			■
MP3 CDs		■				■			■
"Rips" from CDs									
MP3		■		■		■	■		■
RealNetworks		■		■					
Windows Media	■					■	■		
Plays digital audio									
MP3	■	■	■	■	■	■	■	■	■
RealNetworks		■	■	■	■				
Windows Media	■	■		■		■	■		
Streams digital audio									
MP3	■	■	■			■	■	■	■
RealNetworks		■	■		■[1]				
Windows Media	■					■			
Plays digital video									
MPEG	■	■	■			■		■	
QuickTime								■	
RealNetworks		■	■						
Windows Media	■					■			
Streams digital video									
QuickTime								■	
RealNetworks		■	■						
Windows Media	■					■			
Music library	■	■		■		■			■
Copy to portable device	■	■		■		■			■
Artist information	■	■		■	■	■	■		■
On-line "media center"	■	■	■	■	■	■			

Note: 1) Plays streaming audio from Spinner's Web servers only.

Source: Companies' Web sites, product help files, inspection of products.

apparent that WMP would not be a single piece of "Middleware" under the non-settling States' general definition, but a collection of individual functions (and their code) defined by APIs and other interfaces. Given that the high degree of interdependence among many of these features, presumably it would be necessary to redesign the WMP into at least 14 different pieces of "Middleware" (each covering one of the features listed in the table) that could be assembled

into 16,000 variants of WMP, based on the 2^{14} possible combinations (or omissions) of those pieces.

4. Disclosure Requirements Extend Beyond Windows Client Operating Systems.

148. I have focused my attention on the ways in which the non-settling States' definitions of "Middleware" and "Microsoft Middleware Products" apply to Windows PC operating systems (*e.g.*, Windows ME, Windows 2000 Professional, and Windows XP). Although it is hard to fathom any technical justification for the operative breadth of those definitions, a close reading of the non-settling States' Proposal indicates that the two terms apply to far more than Microsoft's desktop operating systems—and may in fact apply to virtually all of its software products. "Network operating systems," for example, which Dr. Ledbetter's testimony suggests means any operating system used on a server,[132] are explicitly defined as "Middleware." The non-settling States also include various applications that run on servers, such as Exchange (for email and related functions), in their list of "Middleware." Although these types of "Middleware" are not subject to the unbinding requirements of Provision 1, they fall under the disclosure requirements of Provision 4.

149. Indeed, the scope of Provision 4 goes considerably further than "network" operating systems, applications, and other software might suggest, because the Provision applies to *any* communication between a Microsoft "program" (which presumably means *any* software distributed by Microsoft) and *any* "Microsoft Platform Software" running on another "computer." The "computer" need not be a PC—servers, handheld devices, and settop boxes are listed explicitly, "without limitation."

[132] Direct Testimony of Dr. Carl S. Ledbetter, March 27, 2002, ¶ 38.

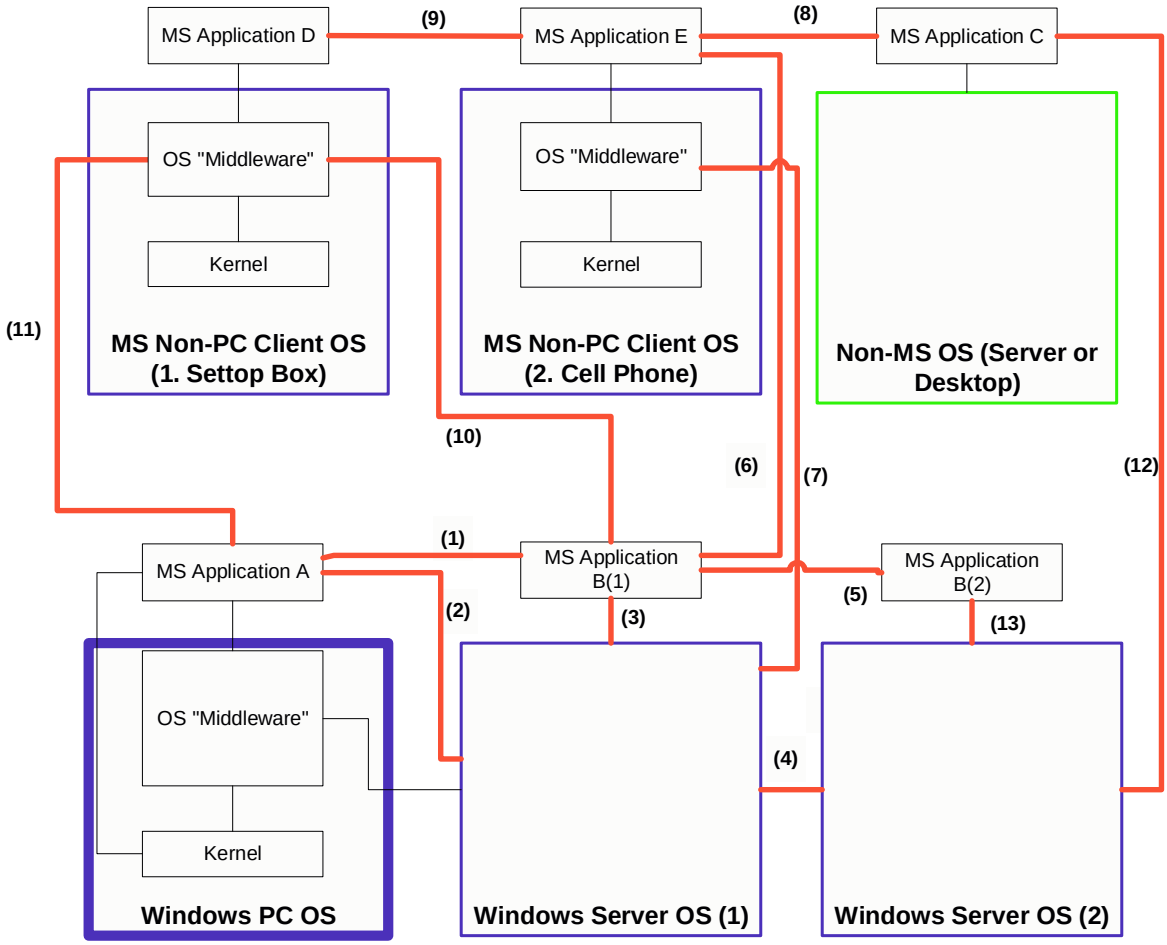


Figure 8. Examples of "Communications Interfaces" that Would Have to be Disclosed Under Provision 4

150. Figure 8 illustrates how this part of Provision 4 would apply and just how broad the sweep of Provision 4 is. It shows six computers, only one of which is a Windows PC (the computer shown in the lower left-hand corner with heavy shading). The other five are: two Windows servers, two non-PC clients with MS operating systems (*e.g.*, one a TV settop box, the second a cell phone), and a computer with a non-Microsoft operating system (*e.g.*, a Linux server or a Macintosh desktop machine) running a Microsoft application. The lines between the various computers represent lines of communication—which could be through APIs or "communications interfaces." I have *not* tried to show all of the possible connections, nor any

of the connections within the many different individual pieces of "Middleware" within each operating system. The heavy red lines illustrate connections that Microsoft would have to document that do not involve any of its PC operating system products.

151. Under the literal terms of the disclosure requirements, Microsoft would have to disclose the detailed interfaces at each end of the interaction and the details of any proprietary protocols used between Microsoft products, even if neither one were a PC operating system or PC "Middleware." Microsoft would have to document every detail of exchanges of information between, for example:

- a Microsoft application running on a PC (*e.g.*, Outlook) and its counterpart running on the server (*e.g.*,. Exchange), as illustrated by the connection labeled "(1)" in Figure 8;
- two copies of a Microsoft server application (*e.g.*, Exchange) running on different servers and synchronizing shared information (connection #5);
- two copies of a component of a Microsoft server operating system (*e.g.*, Active Directory) running on different computers and replicating copies of common information (connection #4);
- "Middleware" in a settop box exchanging information with a Microsoft application running on a server (connection #10);
- an application running on a cell phone obtaining information from a Microsoft application running on a server, which need not even be a Microsoft server (connection #8); and
- a "Middleware" component of a non-PC operating system (*e.g.*, a cell phone) obtaining information from a server (connection #7).

This list is by no means complete, but it suggests that Provision 4 would apply to virtually any Microsoft product that communicates with another Microsoft product. Without meaning to ignore or minimize the issues raised by the scope of the Provision, in general my discussion of Provision 4 below will focus on the Provision's application to PC operating systems.

B. The Requirement to Document "All" Interfaces Has Many Costs.

152. Because of the granular definition of "Middleware," the disclosure requirements could be used by competitors to force Microsoft to "document" (that is, describe in technical detail) countless internal interfaces that are not currently documented APIs. As noted earlier, operating systems and other complex software generally include many interfaces that are not made available as APIs. Although I am not aware of any estimates of the number of such undocumented interfaces in Windows, based on my own programming experience I am confident that it is a substantial multiple of the number of documented APIs.[133] Similarly, Dr. Ledbetter has testified that Novell's NetWare server operating system contains many internal, undocumented interfaces.[134] And David Richards of RealNetworks acknowledged that there are thousands of internal interfaces in RealOne Player that RealNetworks does not publicly document.[135]

153. Exposing more interfaces can increase the functionality to which third-party developers have access, although often it merely duplicates functionality already available through more carefully protected public APIs. Against that benefit, however, platform vendors must weigh several important costs, many of which have been acknowledged by witnesses for the non-settling States:

1. *Stability and reliability.* Some internal interfaces pose severe risks to other programs or the operating system itself if used incorrectly. In some cases, interfaces of this sort have a public version that checks for dangerous parameter values and then passes on the request only if it was formulated correctly and safely.[136] On the other hand, internal

[133] In his testimony, Dr. Appel notes that complex software has many internal interfaces. (*See* Direct Testimony of Dr. Andrew W. Appel, March 13, 2002, ¶ 37.)

[134] Trial Transcript, March 27, 2002, a.m. session (Ledbetter Cross), p. 1552.

[135] Trial Transcript, March 20, 2002, p.m. session (Richards Cross), pp. 736-738.

[136] For example, the use of "pointers" can speed up execution considerably, but runs the risk of overwriting parts of memory in use by other applications or the operating system.

calls may go directly to the unprotected version in the interest of obtaining faster performance. As Dr. Ledbetter has testified in these proceedings, there are several reasons why Novell keeps NetWare's thousands of internal interfaces secret: "There can be several causes for doing that. And they could range from concerns about the level at which we use the interface and the pitfalls of somebody using that interface without sufficient understanding of the operating system. And there are reasons why we might choose not to make the interface available because of security concerns that permitting the interface in an open way would subject the system to certain kinds of hacks from outside."[137]

2. *Flexibility and innovation*. Once an interface is exposed and documented, developers will use it. It then becomes very difficult for the platform vendor to change it. Mr. Green of Sun referred to published APIs as a "commitment" or a "contract" between the system vendor and developers that the APIs will be stable.[138] During his cross examination, Mr. Borduin of Autodesk also spoke of APIs as "implied contracts."[139] Microsoft devotes a great deal of effort to ensure backward compatibility by not changing documented APIs in ways that will prevent them from functioning with older applications.[140] Internal interfaces, however, can be and are changed relatively freely because only other parts of the operating system depend on them, and those parts can be modified and tested thoroughly before release.

3. *Protection of intellectual property*. Some internal interfaces represent valuable intellectual property that has been developed at considerable cost. Documenting those

[137] Trial Transcript, March 27, 2002, a.m. session (Ledbetter Cross), pp. 1555-1556 (quoting Deposition of Dr. Carl S. Ledbetter, February 10, 2002, pp. 21–22).

[138] Trial Transcript, March 18, 2002, p.m. session (Green Cross), pp. 174-176 ("What I mean by the public APIs is a commitment to developers that if they use those APIs in creating their applications, if they write to those definitions published—created by those APIs, that they can be assured that their application will work on subsequent revisions. And this is not exclusive to Microsoft. This is a general industry definition of a public API.") Also in his deposition, Mr. Green referred to the "commitment" as a "contract." (*See* Trial Transcript, March 18, 2002, p.m. session (Green Cross), pp. 174-176 (quoting Deposition of Richard Green, February 15, 2002, p. 163).)

[139] Trial Transcript, April 18, 2002, a.m. session (Borduin Cross), p. 4199.

[140] Some changes can be made to an existing API while maintaining backward compatibility. For example, new, optional parameters can be added so that third-party developers can use them Older applications that do not include those new parameters in their calls of the API will simply use the default values for those parameters. More fundamental changes, however, are likely to cause older applications to fail.

> interfaces precisely enough to make them usable by third-party developers would effectively disclose to competitors how to duplicate that functionality in their own products. As noted earlier, commercial software companies typically rely heavily on trade secrets to protect their intellectual property.

For these reasons (among others), all software has interfaces that are not exposed to the outside, or that are exposed but not documented. Microsoft is unusual not for the number of "hidden" interfaces in its software, but for the opposite—the large number of interfaces it documents and makes available to third-party developers. This is one of the reasons that it is popular with software developers.[141]

154. Developers have various means of discovering undocumented interfaces, but platform vendors almost universally discourage their use. Sun, for example, has warned developers not to use any interfaces in Java Virtual Machines (including Sun's own JVM) other than those published as Java specifications. According to Sun, the use of unpublished interfaces could render the software application in question unable to run on other JVMs and perhaps unable to run on future versions of that particular JVM.[142]

155. My understanding is that, for precisely the reasons listed above, Microsoft's policy is that applications designed for Windows use only documented operating system APIs.[143] Although I understand there may have been instances in which Microsoft's own developers did not follow this policy, I am not aware of any cases in which this has been shown to have given Microsoft's applications a significant advantage or to have prevented other vendors' applications from working with Windows.

---

[141] Direct Testimony of Scott Borduin, April 3, 2002, ¶¶ 35-38.

[142] Sun Microsystems, "100% Pure Java Cookbook," Copyright 1998 (DX 2014 in the liability phase).

[143] Deposition of James Allchin, February 13, 2002, pp. 236-238; Deposition of Henry Brian Valentine, January 29, 2002, pp. 36-41.

156. In light of the non-settling States' sweeping and granular definition of "Middleware," however, it seems certain that Microsoft would have to expose and document thousands of interfaces that are now used only by other parts of Windows. The costs are likely to be high in terms of reduced stability and reliability, reduced flexibility to innovate, and lost intellectual property. In addition, as anyone who has ever tried to document software for use by others knows, the cost of documenting tens or hundreds of thousands of additional interfaces would be significant.[144]

C. The Non-Settling States' Proposal Requires Disclosing Far More than "Interfaces."

157. The loss of intellectual property under the non-settling States' disclosure requirements would be exacerbated by the fact that the requirements go well beyond providing information on the interfaces themselves.

1. The Non-Settling States Define "Technical Information" Broadly.

158. "Documenting" an API normally means providing the information needed by a developer to make use of the service(s) in question. As I discussed earlier, how the service is actually provided—how one goes from inputs to outputs—is normally not disclosed, since the information is not needed by the application developer. It is the platform vendor's responsibility to make sure the API works as advertised, but the actual implementation remains the vendor's intellectual property.

159. The non-settling States' Proposal, however, clearly goes beyond the information needed to use an API to include information that would be useful to other firms seeking to

[144] For example, when I worked on a team at IBM that developed what became VM370, there were 12 developers on the team, but the documentation group had 20 writers. Mr. Borduin of Autodesk also testified that documentation is a "nontrivial process" because it has to be done "carefully, thoroughly." (*See* Trial Transcript, April 18, 2002, a.m. session (Borduin Redirect), p. 4216.)

clone functionality. This comes out most clearly in the statements of competitors who have testified in favor of the non-settling States' Proposal.[145]

160. The non-settling States' Proposal requires that Microsoft disclose the kind of information normally associated with cloning, rather than using, interfaces. (It bears repeating that "cloning" is not in any sense limited to the actual "copying" of code. As Dr. Shapiro and others have testified, a "clone" includes a piece of software that replicates the *functions* of another piece of software, even if that replication is accomplished by something other than the literal repetition of the same source or binary code.[146]) The original proposal explicitly defined "Technical Information" as including information needed for "using and/or implementing" APIs.[147] The revised proposal drops "implementing," but still specifies that "Technical Information includes but is not limited to reference implementations" and a host of other details.[148]

161. A "reference implementation" generally means the source code for implementing a feature on a particular platform. In his deposition, Richard Green from Sun suggested that a "reference implementation" may be a working prototype, rather than the final code used in the production version.[149] However, in the case of the vast number of APIs implicitly defined as "Middleware" by the non-settling States' Proposal, many of which have been in Windows for many years, there would be no reason for Microsoft to have such prototypical code. In such cases, presumably, the only practical approach to the requirement

[145] *See* examples in ¶ 163, *infra.*

[146] Trial Transcript, April 11, 2002, p.m. session (Shapiro Cross), pp. 3475-3479.

[147] Plaintiff Litigating States' Proposed Final Judgment, December 7, 2001, § 22.nn.

[148] Plaintiff Litigating States' First Amended Proposed Final Judgment, March 4, 2002, § 22.nn.

[149] Deposition of Richard Green, February 15, 2002, pp. 262-263.

that a "reference implementation" be provided on demand would be to turn over the Windows source code itself.[150]

162. Even if the source code (prototypical or production) is protected by copyright, the ability to inspect it gives competitors a major leg up in replicating its functionality. In my experience, most of the value from developing complex software derives not from the specific code (which can be written by any competent programmer), but from innovations in architecture, algorithms, and data structures. In this case, Microsoft would have even less protection than usual, because Provision 15 requires that Microsoft license (at zero royalty) any intellectual property that another firm says it needs to exercise any of its rights under the Final Judgment.[151] Determining what a developer "needs" to accomplish a specific goal is highly complex and, in many cases, subjective. In light of the non-settling States' expansive definition of "interoperate" and the requirement in Provision 21 that all provisions be interpreted "broadly," it would be difficult for Microsoft to deny requests for licenses to much of its source code. I am not a lawyer, nor do I have a detailed knowledge of intellectual property law. But as a computer scientist and a professor in a business school, it is clear that the cumulative effect of these provisions would be to strip Microsoft of most of the fruits of its research and development.

163. The revised proposal from the non-settling States also adds language in several places referring to information needed for interoperability between another firm's products and "applications for Microsoft Platform Software."[152] This addition appears to be designed so that vendors of operating systems or other platform software can obtain enough information to

[150] Deposition of William Henry Gates III, February 19, 2002, pp. 309-313.

[151] Plaintiff Litigating States' First Amended Proposed Final Judgment, March 4, 2002, § 15.

[152] *See id.*, §§ 4, 15.

modify their platforms so that applications written to run on Windows will run on ("interoperate with") their platforms.[153] Achieving that ability would almost certainly require Microsoft to license code (royalty free) for many of its Windows operating system APIs, as applications written for Windows are unlikely to work if the other platform does not support all of the Windows APIs the application uses in precisely the same manner as Windows. It is clear from the testimony of several witnesses for the non-settling States that they regard cloning of APIs as a valid use of information gained through disclosure. For example, during his cross-examination Mr. Schwartz of Sun acknowledged that the APIs exposed by Microsoft's .NET framework are documented for developers to use, but complained that "the information necessary to create a competitive implementation of the .NET framework on non-Windows platforms is not available."[154] In other words, Mr. Schwartz complains that while Microsoft shows developers how to use the APIs in its new framework, it does not disclose all of the information that Sun and other platform competitors need to clone them easily. Contrary to Mr. Schwartz's assertion, I note that the open-source "Mono" project reports making good progress in creating a Common Language Runtime (CLR)—part of the .NET framework—that is compatible with Microsoft's, without access to proprietary information from Microsoft.[155] Even if it were true that competitors could not create their own implementations of CLR, however, that would hardly be unusual; software companies rarely disclose all of the information needed

[153] Deposition of Dr. Andrew W. Appel, March 13, 2002, pp. 258-259:

"Q: The purpose [of section 4(a)(i)] being to create an alternative to Windows for running 32 bit Windows applications?

"A: That's right. And for writing alternative to Microsoft Middleware for supporting the applications that would run on top of the Microsoft Middleware."

[154] Trial Transcript, April 9, 2002, a.m. session (Schwartz Cross), pp. 2860-2861.

[155] "Mono," April 11, 2002, http://www.go-mono.com/ (downloaded April 12, 2002).

to clone their newest products. For example, Dr. Ledbetter testified that Novell's email client and server software use undisclosed communications protocols to share information.[156]

2. The Internet Explorer and Office Provisions Explicitly Require that Microsoft Convey Valuable Intellectual Property.

164. In addition to the disclosure requirements that apply to Microsoft "Middleware" in general, two of the provisions explicitly require that Microsoft convey its valuable intellectual property. Provision 12 requires that Microsoft license on an open-source basis "all source code for all Browser software." It is not clear what code would be covered by this requirement, as the non-settling States' Proposal defines "Browser" only in terms of "IE 6" and MSN Explorer 6.1" (and their successors).[157] My understanding is that the latter is primarily a "shell" designed to make it easier to use MSN online services (including email). What constitutes IE code has, I understand, been an unresolved issue since the 1997 proceedings in which Microsoft was alleged to have violated a 1994 consent decree, and the non-settling States' Proposal provides no definition. In light of the integrated and interdependent nature of Windows, in which the same code serves both Web browsing and other functions, there is no clear way to define that code if it is intended to encompass the functionality itself and not just end-user access to Web-browsing functionality. Whatever the ultimate definition of the affected code, under the non-settling States' Proposal any party (including competitors) would be free to use it in modified or unmodified form for any purpose. Moreover, any innovations in Web browsing software made by Microsoft over the next 10 years (or more) would have to be given away.

[156] Trial Transcript, March 27, 2002, a.m. session (Ledbetter Cross), p. 1601.

[157] Plaintiff Litigating States' First Amended Proposed Final Judgment, March 4, 2002, § 22.e.

165. There is less ambiguity concerning the code covered by Provision 14, regarding Office, although the code is still not fully specified. At a minimum, it includes all of the source code compiled to produce Microsoft Office for both Windows and the Macintosh. However, the provision also requires Microsoft to license "all parts of the source code of the Windows Operating System Product necessary for the porting" of Office to other platforms. That requirement might well include the source code for any Windows APIs relied upon by Office (there are surely many) that do not have near-exact equivalents in the operating system(s) to which Office would be ported.[158] Thus, the requirement could extend well beyond Office itself. Moreover, as with the "Browser" provision, the obligation extends to any innovations incorporated into Office (or APIs on which it depends) over the next decade.

VI. The "Unbinding" Provision in the Non-Settling States' Proposal Is Infeasible.

166. The first provision of the non-settling States' Proposal requires that, for every Windows Operating System Product sold by Microsoft, it offer an "unbound" version "from which the binary code for each Microsoft Middleware Product (including any code providing similar functionality that has been included in any other Microsoft Middleware Product) may be readily removed such that this "unbound" Windows Operating System Product performs effectively and without degradation (other than the elimination of the functionalities of any removed Microsoft Middleware Products)." Given the non-settling States' broad and highly granular definition of "Middleware Products," this requirement is probably impossible to meet. Even if it were to prove technically feasible, the costs would be great for end users and third-

[158] In his cross examination, Professor Shapiro appears to believe that it would be inappropriate to facilitate such cloning of APIs, but could not identify how the existing proposal foreclosed such use. (*See* Trial Transcript, April 15, 2002, a.m. session (Shapiro Cross), pp. 3576-3582.)

party developers as well as for Microsoft. It confuses "components" with "independence," and takes theoretical ideals of "modularity" to new and unrealistic extremes.

A. Modules, Components and Interdependencies: the Automobile Analogy

167. Windows, like all complex software (and most industrial products) is "modular": it is built from subassemblies on which smaller teams work and which are then assembled into the final product. It also has many functional "components": subsets of its functionality are accessible through APIs. As I noted earlier in section V.A.1, the fact that an operating system or another complex piece of software is built from many different modules of code should not be confused with the notion that individual modules can be removed from the system without affecting other functionality. During his cross-examination, Mr. Borduin of Autodesk made exactly this point in response to questions about modularity:[159]

> When we talk about modular, however, you have to—you have to think about why, you know, in which ways is the software modular. In other words, is it modular in order to meet a set of requirements for facilitating further development of the product? And that's the sense in which many of my own products are modular.
>
> And then there's also modular in the sense of actually being designed to allow code to be moved around or to allow pieces to be swapped in and out, and those aren't necessarily the same thing at all from a software design perspective.

168. There is no simple correspondence between the functional components exposed by APIs and the physical modules of code. One functional component is likely to rely on several modules of code, and the same module of code is often used by multiple functions. As a result, there are substantial interdependencies, many of which are quite subtle and hard to anticipate. Anyone who has worked with computers for long can recount experiences in which one change caused problems in a seemingly unrelated part of the system. For example,

[159] Trial Transcript, April 18, 2002, a.m. session (Borduin Cross), p. 4169.

changing the software "driver" for the video display may affect printing, which has its own software drivers. To a large extent, the tens of thousands of "computer viruses" that have been identified also illustrate the difficulty of predicting the consequences of these interdependencies—no operating system developer deliberately intends to allow a virus to occur.

169. To deal with these issues, it is not enough to make sure that individual blocks of code are error-free; one must also test the complete system extensively with different combinations of hardware and applications software.[160]

1. Modularity and Interdependencies in Automobiles

170. Automobiles provide a useful, though far simpler, analogy that makes clear why modularity does not eliminate interdependencies, and indeed may increase them.[161] Ever since Henry Ford invented the modern assembly line, cars have been built up from sub-assemblies. These sub-assemblies or modules are put together to create the car itself. As cars have evolved, components have become more complex, integrated, and interdependent.

171. These interdependencies can be subtle and circular, direct and indirect. In the earliest cars, for example, headlights were self-contained, powered by their own batteries or by kerosene, and were sometimes sold as options. It was easy to substitute a radically different headlight design. Likewise the engine was a separate system, usually started by a manual crank.

172. Modern cars have electric starters—which require a powerful central battery, which also powers the headlights. The central battery, in turn is dependent on the alternator for recharging, which is driven by the engine, which cannot start without the starter motor, which

[160] Deposition of Robert Short, February 8, 2002, pp. 83-85; Deposition of Henry Brian Valentine, January 29, 2002, pp. 343-345.

[161] Cars are simpler than complex computer systems in many ways, one of which is that the interfaces between components tend to be simpler and more obvious upon inspection: one part may be bolted to another, with a rotating shaft as the primary interface.

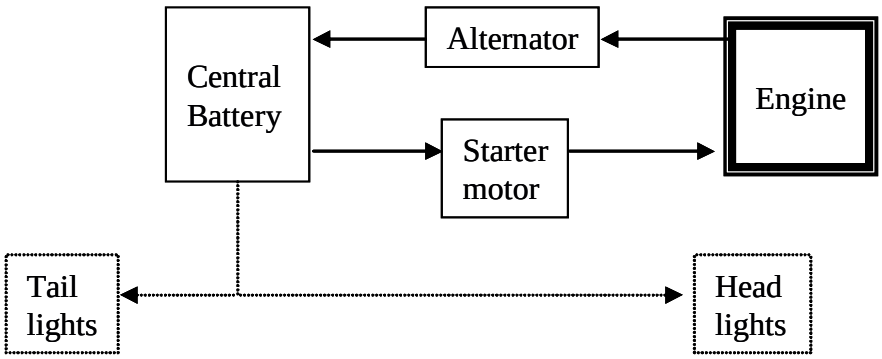


Figure 9. Complex and Circular Interdependencies in Automotive Electrical System

is powered by the battery, as illustrated in Figure 9. Although each of these modules is physically separate, if any one of them is removed the system will fail. We can now see why the dependencies between modules are far more complex than illustrated earlier in Figure 7 (p. 66): is the alternator dependent upon the engine (a direct dependency) or is the engine dependent upon the alternator (an indirect dependency)? In fact, they are dependent on each other, a phenomenon I call "circular dependency." I think almost everyone would view the modern car design to be much superior and innovative than the early car, even though it does have many more interdependencies. These same innovations have also occurred in modern operating systems, such as Windows, though most components of complex software are more like an automobile's engine—designed for a specific family of automobile models and extremely difficult to replace with another manufacturer's component—than like relatively standardized automobile batteries.

2. Finer Grain Interdependencies: Automobile Audio Systems

173. Audio systems in cars provide a more recent illustration of growing and more finely grained interdependence. Twenty years ago, it was easy to remove the radio in most cars; the only loss in functionality was the ability to listen to broadcasts. In many current cars, however, the radio is part of an audio system. A single case contains a tuner for broadcasts, a tape player, a CD player, and an amplifier and controls shared by all three functions. One

cannot remove the "radio" without removing the tape and CD players as well. Replacing any one of those functions requires replacing all three with a new, integrated unit that supplies those functions, or a subset of them. And these audio components are dependent on other "modules" in the car, including speakers and the battery, which expands the web of interdependencies described above.

174. As automobiles become ever more sophisticated, these interdependencies continue to increase. Additional features are being integrated into car audio systems, creating additional interdependencies. For example, "hands-free" cellular phone options in some models work through the audio system. In other models, controls for computerized navigation systems are integrated with the audio system or climate controls.

B. Software Componentization Does Not Necessarily Reduce Interdependencies.

175. As with car audio systems, the more componentized an operating system is, the more interdependencies there are likely to be, and the harder it may be to extract modules without affecting other functions as well. That is because some modules are used in the provision of several functions (much as the case, amplifier, and controls in an integrated car tuner/tape/CD player are used for several functions). For example, operating systems that use a stand-alone application (such as Navigator) for Web browsing can "delete" the Web browser without affecting other functions, and it is easy to install a different Web browser to restore the lost functionality.[162] In Windows, by contrast, Web browsing functions are provided by numerous modules of code, many of which also serve other parts of the operating system or third-party applications. Thus, while it may be easy to remove the small file (the blue-shaded

[162] Stand-alone Web browsers are also built from many smaller modules, but may not be designed to expose functional components through published APIs that other applications can use.

box labeled "Browser UI" in Figure 7 on p. 66) that launches the Web browser when the user double-clicks the IE icon, removing the code that renders HTML text would disable other operating system functions, such as the help system, that use the component, as well as third-party applications that use it.[163] Replacing each of those components with substitutes from third-party "Middleware" vendors would be extremely difficult, if not impossible, because there are too many interdependencies for it to be practical without replicating large portions of the deleted code in considerable detail. On the other hand, it is relatively easy to install a third-party stand-alone Web browser for the end user to employ in surfing the Web, because there are far fewer interdependencies with other software and the "calling program" (the human user) is far more adaptable and flexible than most software.[164]

176. Similar but different interdependencies exist in other operating systems. For example, the Mac OS X relies on the Portable Document Format (PDF) in a much more integrated way than does Windows. The Quartz "layer" in Mac OS X includes code and associated APIs to let developers "easily embed and manipulate PDF data within any Mac OS X application."[165] Removing the "Middleware" modules in Mac OS X that support PDF would disable important functionality beyond what one might expect. In contrast, Windows does not include built-in support for PDF; users obtain separate application software, most likely from Adobe, which developed PDF and owns associated intellectual property.

---

[163] Microsoft Corporation's Response to Plaintiff Litigating States' First Set of Interrogatories in Remedy proceedings, January 11, 2002, pp. 27-28. The "stub" file that launches "IE" does not actually provide the user interface; it merely launches a more general user interface module (Shell Doc Viewer) with the appropriate parameters.

[164] The car analogy is again useful. It is relatively easy for a rental car company to substitute one type of car for another (especially if the substitute normally is more expensive); human drivers are fairly adaptable. However, it would be hard, and require many other modifications, to substitute the engine in one of those cars with the engine from the other.

[165] Apple, "Mac OS X: An Overview for Developers," February 2001, p. 5, http://developer.apple.com/macosx/pdf/macosx_overview.pdf (downloaded April 3, 2002).

C. "Removing" Code Is Far More Problematic than Removing End User Access.

177. Under the Proposed Settlement negotiated with the United States and the settling States, Microsoft must make it easy for OEMs and end users to remove direct access (the main user interface) to certain defined types of middleware.[166] In terms of Figure 7 (p. 66), they can disable the blue-shaded user interface blocks of code, thus blocking normal end-user access. That is essentially what the "IE removal" program developed by Professor Felten during the liability phase did. It removed the small executable file (IEXPLORE.EXE) that users normally use to launch IE and changed some other settings.[167] As a result, users could no longer browse the Web using IE (except in very limited circumstances), but no other parts of Windows (such as Help) or third-party applications (such as Intuit's Quicken or RealNetworks' RealOne Player[168]) that depend on components associated with IE would be affected. My understanding is that Microsoft is currently developing a version of Windows XP for release later this year that will offer OEMs and end users the ability to uninstall or later reinstall end user access to the set of middleware functions defined in the Proposed Settlement.[169] The actual code providing the "middleware" functionality is likely to be affected little, if at all.

178. Meeting the requirements of the anti-binding provision of the non-settling States' Proposal, however, would require far more drastic action. Microsoft would have to remove not only end user access to "Middleware," but also all of the code that supports the

---

[166] Second Revised Proposed Final Judgment, February 27, 2002, § III.H.1.

[167] Trial Transcript, December 14, 1998, a.m. session (Felten Cross), pp. 42, 68, 71-72. *See also*, Direct Testimony of Jim Allchin, January 27, 1999, ¶ 163 ("[Dr. Felten's] version of Windows 98 is 99.93% as large as the Microsoft version of Windows 98."). Felten's program also *added* code to Windows so that another browser could assume some duties that were normally assigned to IE. However, my understanding is that the Court of Appeals ruled that it was not anticompetitive to have IE launch in those limited circumstances.

[168] "Quicken 2002 for Windows: Installing or Uninstalling the MS Internet Explorer Browser from the Quicken CD," http://www.intuit.com/support/quicken/2002/win/1941.html (downloaded March 26, 2002); Trial Transcript, March 20, 2002, a.m. session (Richards Cross), pp. 626-633.

[169] Deposition of Steven A. Ballmer, February 8, 2002, pp. 292-294.

associated functionality. Because of the integrated nature of Windows, that requirement is much more ambiguous and harder to implement. As noted earlier, for example, the Proposal does not define the code that is encompassed by IE 6. Clearly the code in question consists of more than the small file that launches the browsing function when the user double clicks the IE icon, but it is hard to know where to stop.

179. In terms of Figure 7 (p. 66), how many modules are "IE" and hence must be removed? As soon as one moves below the blue-shaded module, tough choices emerge, because most of the modules supporting the Web browsing functionality also support other functions. In terms of the wall analogy, it may be relatively easy to remove a block from the top of the wall, but once one starts removing the blocks below it, other parts of the wall are likely to fail. The non-settling States' proposal is even riskier, because its granular definition of "Middleware" means that the wall must be designed so that individual blocks many layers down in the wall are removable. And operating systems are far more fragile than block walls, as discussed before, because the dependencies are not one-way and developers depend on APIs exposed by blocks of code that are throughout the system.

D. Meeting the "Unbinding" Requirements Would Require Completely Redesigning Windows and Sacrificing Quality and Performance.

180. One could, of course, redesign Windows to make it easier to remove certain components, just as one could make it easier to remove a car's radio or other audio components individually by putting the tuner, tape player, and CD player in their own cases, each with its own amplifier and controls. (This assumes that one need not remove the speakers and battery to "remove" the radio—but if they are defined to be part of the "radio," they would have to be duplicated too.) Professor Appel in his trial testimony, and Assistant Attorney General Greene

of California during his deposition, suggested something similar for Windows.[170] Under what I take to be their approach, if a module were used by "Middleware," the code would be duplicated each place it was used. That way, an OEM or Third Party Licensee, they say, could delete both the code itself and each piece of "Middleware" with which it is associated, yet still have duplicate copies of the code be available for other operating system functions. However, the copy used for internal, non-"Middleware" use would not be made available to developers through APIs (since one of the non-settling States' goals in forcing code removal is to stop developers from relying on Microsoft's APIs).[171] Duplicating the implicated code for only a modest number of functions would be technically feasible, although it would increase the size of Windows and probably reduce its performance and reliability. However, it would clearly be infeasible in any reasonable time frame to build a working version of Windows that allowed licensees to remove any combination of the countless functions that could be called "Middleware" under the non-settling States' definition. Moreover, such a version would be much larger, much less efficient, and much harder to maintain and update than the current integrated version, all of which would raise costs substantially. And, in the quest to remove certain code, multiple copies of it would be added to the operating system.

1. HTML Rendering as an Example

181. The non-settling States' Proposal requires Microsoft to redesign Windows so that one type of functionality can be removed without impairing any other functionality.[172]

[170] Trial Transcript, April 9, 2002, p.m. session (Appel Cross), pp. 2993-2995; Trial Transcript, April 10, 2002, p.m. session (Appel Redirect), pp. 3208-3210; Deposition of James Thomas Greene, March 12, 2002, pp. 33-35.

[171] Deposition of James Thomas Greene, March 12, 2002, pp. 39-40.

[172] Plaintiff Litigating States' First Amended Proposed Remedy, March 4, 2002, § 1.

Thus, if a licensee removes the Web "browser,"[173] all of the Help system, the file manager (Windows Explorer), Windows Media Player, and the email client (Outlook Express) should still work. But all of those functions (and others) depend on many of the same files the "browser" relies on. In particular, all use a module (shdocvw.dll, or "Shell Doc Viewer") to draw parts of the interface. Moreover, for some purposes, all also rely on the HTML renderer (mshtml.dll) "Removing" the key binary files for the "browser" would disable all of those other functions as well.

182. Figure 10 illustrates the issue in a highly simplified way that does not include the numerous other blocks of code involved. I show six Windows functions that rely on Shell Doc Viewer; two of them (IE and MSN Explorer) are "browsers," but the others are not. All of these functions also depend on HTML rendering (and numerous other modules, some in common, some not). I also show two third-party modules relying on Shell Doc Viewer and HTML. Application 1 is using primarily "Web browsing" functionality (*e.g.*, Quicken), while Application 2 is a media focused application that relies on Windows Media Player and its components for various purposes (*e.g.*, AOL's Winamp).[174] One way to avoid disabling other parts of the operating system and applications that rely on "IE" components would be to provide an exemption for Shell Doc Viewer and the HTML renderer (along with other modules posing similar problems). That approach would leave the other parts of Windows still functioning and also would continue to make those services available for third-party applications. Those seem to me to be important benefits, but the non-settling States have made clear that such an approach does not achieve the purpose of Provision 1. Their goal, according

[173] The non-settling States define the "browser" as IE and MSN Explorer; presumably a licensee could demand the removal of one but not the other (*e.g.*, AOL might want MSN Explorer removed, but not IE). For simplicity, I have treated the two as a single "browser."

[174] Some of these functions may call the HTML renderer directly, not just through Shell Doc Viewer as shown. I have omitted those possibilities for simplicity.

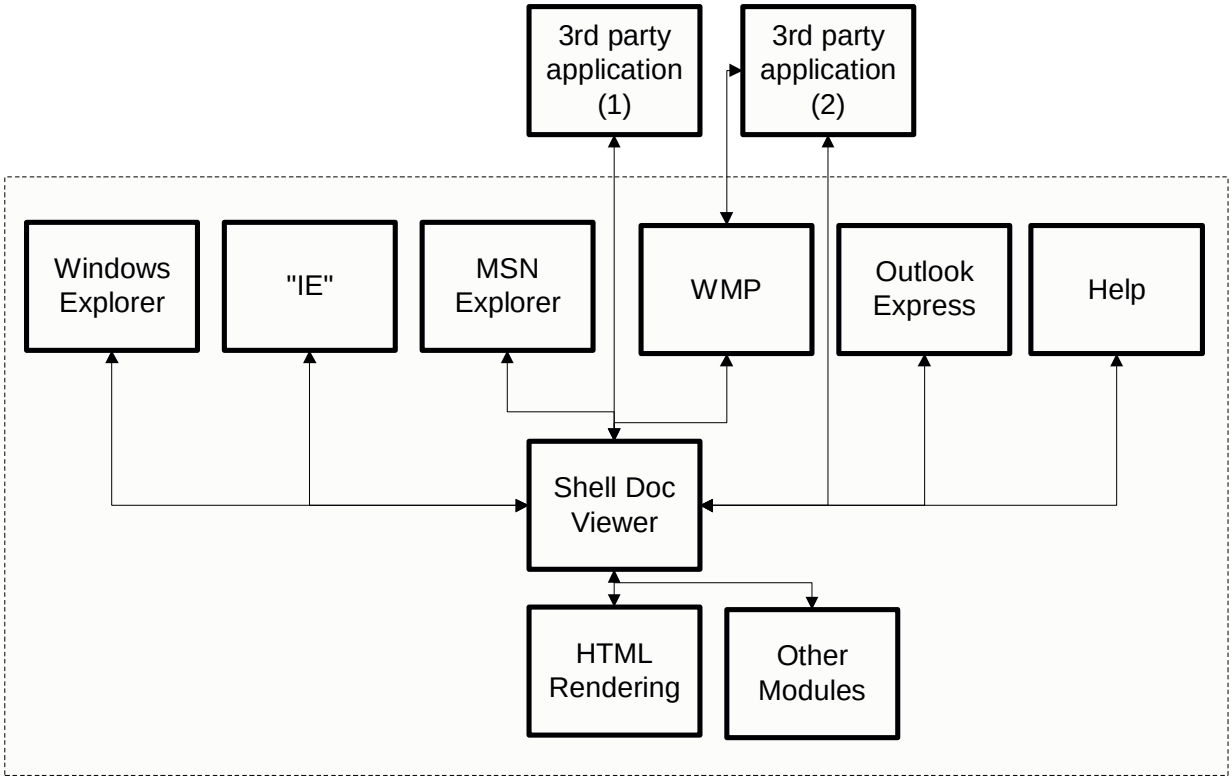


Figure 10. Simplified View of Shell Doc Viewer and HTML Dependencies in Windows

to Assistant Attorney General Greene, is to remove not just the end user's access to the "Middleware," but also the Microsoft code that supports related APIs. Otherwise, they worry, third-party developers will choose to write to Windows rather than to competing "Middleware."[175] Apparently they believe that consumers will be better off by making Windows less functional and dependable for developers.

183. One of the "solutions" suggested by Professor Appel[176] is illustrated in Figure 11. The original Shell Doc Viewer and HTML modules provide services for IE and MSN Explorer, and they also expose the relevant API(s) for other operating-system "Middleware"

[175] Deposition of James Thomas Greene, March 12, 2002, pp. 37-40.

[176] Trial Transcript, April 10, 2002, p.m. session (Appel Redirect), pp. 3208-3210.

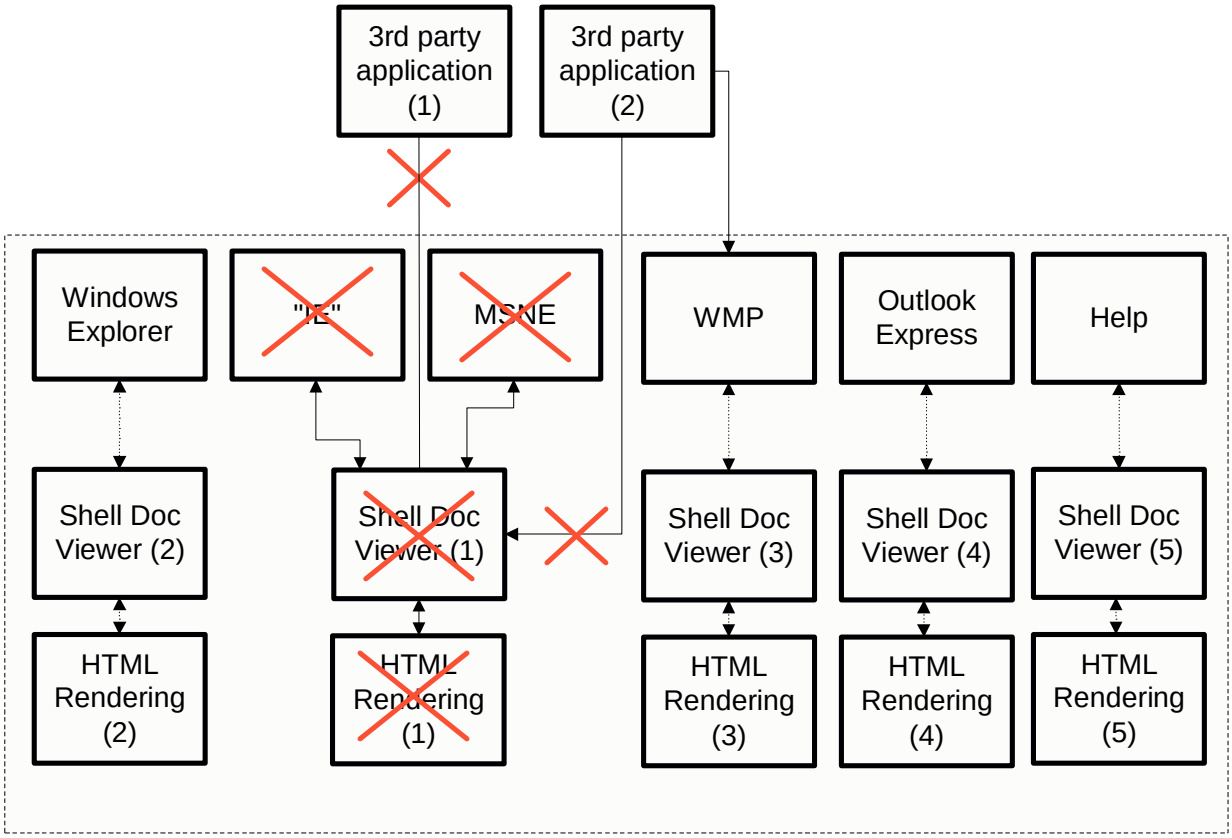


Figure 11. One of Professor Appel's "Solutions" to Simplified Version of HTML Dependencies

and third-party applications. We now add four more copies of those modules (numbered 2-5) for each of the other pieces of "Middleware." (In reality, there are more than four other functions, so more than four other copies would be needed.) With this design, the licensee wishing to delete the "browser" can delete Shell Doc Viewer and the HTML rendering engine and prevent the use of associated functionality by third-party applications, but still maintain "non-browsing" functionality in the operating system (Windows Media Player, Help, and so forth). However, the APIs for those other copies are all converted to internal interfaces (shown by dotted lines) within each piece of "Middleware," so that applications and other "Middleware" cannot call them. To the extent that the non-Middleware components of the operating system rely on either Shell Doc Viewer or the HTML renderer, a single copy of these

could be included in the operating system for joint use by such "core" elements of the operating system, provided that no functions were exposed and documented for use by applications or Microsoft "Middleware."

184. This solution is troubling on several levels. First, the approach results in duplicating code in multiple places in the operating system, rather than repeatedly calling a single piece of code. This violates one of the basic principles of modern software design.[177] Having multiple functions share a single copy of code not only reduces the size of the overall software product (which has some benefits in itself), it also makes it easier to update and maintain the product. By the same token, bug fixes or other improvements in that code need to be made only once, not multiple times, and users need only download and install one corrected file, not many.

185. Second, purposely disabling the ability of third-party developers to use code from Microsoft seems inconsistent not only with common sense, but also with the spirit of the non-settling States' general emphasis on (indeed their proposed requirement) compelling Microsoft to provide complete details on how to use functionality in its platform products. Note that not only is Application 1 denied the functionality, the change also "breaks" Application 2, which is a media playing application and hence one that might not have been expected to be affected.

186. This solution also may be in violation of Provision 4, which requires that Microsoft disclose all "interfaces" for all "Middleware Products," the general definition of which clearly would apply to Help and all of the other functions shown in the figure. (These functions expose APIs for use by third-party applications, they are not part of the kernel, and

[177] Dr. Appel agreed in his testimony that this "solution" could lead to a waste of system resources and stated that this "solution" "is not necessarily the approach I would recommend." (*See* Trial Transcript, April 9, 2002, p.m. session (Appel Cross), pp. 2994-2996.)

there are competing implementations in other operating systems and "Middleware.") If copies of Shell Doc Viewer and the HTML Renderer (which themselves appear to be "Microsoft Middleware Products" under § 22.w.(ii)) are created for use with each of the remaining functions, there will have to be "interfaces" between them and the other functions.

2. The Impossible Dream

187. The concept of a completely modular operating system has been a "holy grail" for computer scientists for decades, one frequently sought but never found. Even in their dreams, however, computer scientists have not imagined a modularity as granular as that embodied in the non-settling States' Proposal. Modularity has generally meant being able to replace major elements, such as the file system or the graphical user interface, not the code supporting arbitrary combinations of individual APIs. And even the far more modest vision has largely failed in practice. Although the idea of building an operating environment with "best of breed" components obtained from various vendors has some superficial appeal, getting them all to work together smoothly requires an enormous effort that creates ongoing support problems.[178] Each time one component changes, it may create incompatibilities with others. When things go wrong, as they inevitably do, it is hard to figure out where the problem lies or which vendor is at fault. Often the fault lies not in any individual component, but in the interactions among them. Mr. Borduin testified to such experiences in connection with an Autodesk Web site that was built using software from multiple vendors. Although the individual components worked well, "the integration of those and the maintenance of those was much more difficult."[179] He went on to testify:[180]

[178] Deposition of Roger Michael Needham, March 1, 2002, pp. 86-88.

[179] Trial Transcript, April 18, 2002, a.m. session (Borduin Redirect), p. 4228.

[180] Trial Transcript, April 18, 2002, a.m. session (Borduin Redirect), p. 4228.

> And we would, for instance, discover that when we installed the next release of a particular component, that the system would break and we would call the particular vendor, describe our technology stack, and they would say, "Well, we've never heard of that combination before," which is another way of saying, "Good luck."
>
> So we, in fact, decided that for our purposes the superiority of the individual components was not sufficient to justify those additional investments and we went back to a much more vertically-integrated Microsoft solution stack for that application.

188. I am not aware of any vendor of operating systems who has even attempted to develop anything remotely comparable to the extremely granular modularity that the non-settling States' Proposal would require.[181]

3. The Unbinding Provision Is Equivalent to Requiring Microsoft To Provide Many Thousands of Versions.

189. In their recent revision, the non-settling States attempted to "clarify" that their proposal would require only two versions of Windows rather than thousands: one would be integrated, as today, and one would "unbind" all pieces of "Middleware" so that they could be removed in any combination by OEMs or Third-Party Licensees.[182] This revision is a verbal distinction without a technical or practical difference. Microsoft would still be required to run tests of all of the variations that licensees might conceivably create by removing various combinations of "Middleware," as Assistant Attorney General Greene acknowledged in his deposition.[183] Although some combinations might not have to be tested individually, as suggested by Dr. Ledbetter, the number of variations that would have to be tested would be

---

[181] Linux and the various flavors of Unix come closest, and distributions of Linux often include more than one piece of "Middlware" to perform the same function. Red Hat Linux, for example, includes both the KDE and GNOME GUIs. Other choices, however, are dictated by which GUI the user chooses, as some applications and various pieces of other "Middleware" distributed on the CD work only with one GUI or the other, not both.

[182] Plaintiff Litigating States' First Amended Proposed Final Judgment, March 4, 2002, § 1. In practice, I suspect that Microsoft would have to offer only one "version" as the States define it. That is because an integrated version would perform so much better than the fully modular, removable one that it would be impossible to satisfy the requirement that the modified versions perform as well as the integrated one in all respects save the omitted functions.

[183] Deposition of James Thomas Greene, March 12, 2002, pp. 46-48.

extremely large.[184] Whether one calls those variations "versions" or "alternative configurations," the practical implications are the same. Indeed, in many ways the new proposal makes the task harder, not easier: Microsoft would be forced to build Windows in such a way that licensees could create their own versions in any conceivable combination, even if no one actually wanted to use them. Moreover, Microsoft would not have the opportunity to make custom modifications of the source code to address problems unique to the removal of particular combinations of "Middleware" that a licensee might demand.

190. The burden of the unbinding requirement applies not only to Microsoft's current and future operating systems, but also to earlier versions, including Windows 2000 and Windows ME. Initially the burden would be limited by the fact that only the "Microsoft Middleware Products" listed in section 22.x(i) would have to be made removable, although all of the basic dilemmas would remain. I doubt even that limited application of the "unbinding" requirement could be implemented in the six months allowed, especially as it would apply to at least four different operating system products—Windows ME, 2000 Professional, and XP Home and Professional—built on two very different code bases, one of which (the one used in ME) is obsolete and is no longer being developed by Microsoft. Moreover, after that first round the number of "Middleware Products" affected would be virtually unlimited for the reasons discussed above.

E. "Removing" Code Generally Serves No Useful Purpose and Is Likely To Harm Consumers and Other Third Parties.

191. The high costs of reengineering Windows to "unbind" all "Middleware" (which include a substantial risk of failure to create a workable product) do not yield offsetting cost

[184] Direct Testimony of Dr. Carl S. Ledbetter, March 27, 2002, ¶¶ 59-61.

savings or technical benefits. Instead, they are likely to generate more costs for consumers and third-party applications developers.

1. Storage Costs for Unused Code Are Minimal.

192. Physically removing code from Windows, rather than simply inactivating access to associated functionality, could reduce the size of Windows for customers that choose to omit large amounts of "Middleware." For those users who omit modest amounts of "Middleware," the size of the system is more likely to grow, however, as increased code to reduce interdependencies (*e.g.*, multiple copies of the HTML rendering software, one for each of the many pieces of Middleware that used that functionality) would more than offset reductions in the numbers of modules. Even if the size of Windows were reduced, however, there will be little gain to consumers.

193. In computing devices with extremely limited storage, such as hand-held devices or appliances that use embedded systems, removal of code for unneeded functions can be valuable because it frees up space for code to provide other, more valuable functions. In today's PCs, however, random access memory (RAM) and disk storage space generally are not in short supply. A business that purchases a new PC today is likely to get 256 megabytes (256 million bytes) of RAM or more and a hard disk with a capacity of 20 gigabytes (20 billion bytes) or more. On the Dell Web site, as of April 3, one could upgrade from 20 to 40 GB of hard disk capacity at an incremental cost of $20, or $1.00 per GB.[185] A full installation of Windows XP Home uses slightly less than 1.6 GB of hard disk space, for a total storage cost of

[185] "The Dell Online Store: Build Your System," http://configure.us.dell.com/dellstore/config.asp?customer_id=19&keycode=6V355&order_code=44PSW (downloaded April 3, 2002).

less than $1.60 at that rate. The potential reduction in storage required from the actual removal of "Middleware" would, of course, be a fraction of that amount.[186]

2. The Impact on Developers

194. In addition to the problems it would create for developing and testing Windows, the prohibition on "binding" would impose large costs on developers that rely on Windows as a platform. And that, in turn, would harm end users. In the short run, users' existing applications would no longer work if any of the APIs on which they relied were not included in the version of Windows purchased from a computer maker or third-party licensee. Representatives from several ISVs—including Mr. Borduin of Autodesk, Heather Davisson of Opus-i, and Brent Frei of Onyx—have testified to the problems that would be created if OEMs and Third Party licensees used their rights under Provision 1 to delete code containing APIs.[187]

195. The non-settling States seem remarkably cavalier about these problems. Assistant Attorney General Greene, for example, agreed that it would just be "tough luck" for an application developer (or the user of that application) if an application would not run on versions of Windows lacking the requisite APIs.[188] I find it telling that the only ISV testifying for the non-settling States is RealNetworks, which competes primarily with one of Microsoft's "Middleware Products."

196. Although it dealt with databases and "application server" software—both of which are platforms for in-house corporate applications—not operating systems, Oracle's decision in 2000 to simplify its product line illustrates the benefits that come from providing a

---

[186] As an illustration of how rapidly prices change in this industry, a visit by my researcher to the same Web site a month earlier yielded a cost almost three times as high.

[187] For a discussion of why these inconsistencies in the Windows platform would be harmful to ISVs, see Direct Testimony of Scott Borduin, April 3, 2002, ¶¶ 53-77; Direct Testimony of Heather Davisson, April 12, 2002, ¶¶ 27-31; Direct Testimony of Brent Frei, April 8, 2002, ¶¶ 3, 33-39.

[188] Deposition of James Thomas Greene, March 12, 2002, pp. 39-40.

consistent platform for developers and users. Prior to that time, Oracle offered its software in a modular way, with customers able to decide which features they wanted to purchase (with up to 75 modules available). The problem was that different customers chose different add-on modules, which made it difficult to support countless different Oracle configurations. Oracle explained its decision to simplify its product line by "bundling" modules rather than selling them separately in part on the grounds that it would make it easier for customers to maintain Oracle software.[189] Note that this product is used by relatively sophisticated IT professionals; the problems of inconsistent platforms are much harder for novice PC users to solve.

197. To the extent that licensees used the non-settling States' remedies to create multiple versions of Windows with differing combinations of APIs, applications developers and consumers would lose one of the greatest benefits of Windows—a platform for applications that supports new functionality while providing backward compatibility for most existing applications. Developers write applications to run on Windows not just because it has many users, but because it offers many useful services to applications in the form of APIs. Moreover, Microsoft provides programming tools and support to make it relatively easy for developers to use those APIs. At present, developers can count on all copies of Windows having the code that supports virtually all of those APIs.[190]

3. Linux and Unix as Cautionary Examples

198. The ways in which applications depend on operating systems are well illustrated by experiences with Linux, where variations in libraries and GUIs lead to incompatibilities

[189] Mary Jo Foley, "Oracle: Have It Our Way," *ZDNet News*, October 3, 2000, available at http://www.zdnet.com/filters/printerfriendly/0,6061,2636467-2,00.html (downloaded January 12, 2002); John Cox, "OpenWorld: Ellison Promises $1 Million If His DMBS Doesn't Perform," *Network World*, October 9, 2000.

[190] There are some differences, with the "Professional" edition of Windows XP, for example, supporting more functions than the "Home" edition.

despite the presence of a common kernel providing core services. Reviews of Linux applications frequently mention the problems that the reviewer had in getting the application to run on specific Linux distributions and how the reviewer solved them by tracking down and then installing the specific libraries ("Middleware") needed for that combination of application and Linux distribution.[191]

199. The experience with Unix over the past 25 years also illustrates the problems that arise when a platform fragments. Unix originally was developed by a single firm, AT&T, but over the years it "forked" into many different versions.[192] There are many different "flavors" of Unix on the market today, which are not fully compatible with one another. Few applications are offered for more than a handful of flavors.[193] Because development costs are amortized over much smaller sales, the prices of Unix applications tend to be higher than for Windows (or the Macintosh), and often they lag in features and ease of use.[194] Based in part on

---

[191] For a time, Linux versions were using two different C libraries, glibc and libc. This caused considerable concern at the time. *See, e.g.*, Andrew Leonard, "Is Red Hat becoming Linux's Microsoft?" *Salon.com. Technology*, July 14, 1999, available at http://www.salon.com/tech/feature/1999/07/14/redhat/index1.html (downloaded January 21, 2002). More recently, see Charles Babcock, "Linux vs. Linux" *Inter@ctive Week*, March 20, 2000, available at http://www.zdnet.com/filters/printerfriendly/0,6061,2470425-2,00.html (downloaded January 21, 2002). It was reported that software that runs on one version of Linux cannot be guaranteed to run smoothly on another.

[192] Peter H. Salus, *A Quarter Century of UNIX* (Reading, MA: Addison-Wesley, 1994), chapter 25, pp. 209–212.

[193] For example, although Netscape Navigator is known for running on multiple platforms, the only Unix flavors for which the latest major release of Netscape (version 6.x) is available are HP-UX and Sun Solaris (plus Linux). Only the version for Linux is supported by Netscape—the other two releases were ported by HP and Sun, respectively, and the HP-UX and Solaris releases lag behind those for Windows. (*See* "Choose Operating System," http://home.netscape.com/download/1126101/10004-en----_qual.html (downloaded January 17, 2002); "Choose Version," http://home.netscape.com/download/1126101/10004-en-win32---_qual.html (downloaded January 17, 2002); "Additional Versions of Netscape 6," http://home.netscape.com/download/N6partners.html (downloaded January 17, 2002); "Netscape 6.1 Early Access (EA) for the Solaris Operating Environment," http://www.sun.com/solaris/netscape/index.html (downloaded January 17, 2002); "Receipt," https://software.hp.com/cgi-bin/swdepot_parser.cgi/transaction_processor/transaction_processor.cgi (downloaded January 17, 2002).)

[194] For example, Corel sells version 8 of its WordPerfect word processing program for UNIX. The program is available for only four versions of UNIX (plus Linux), excluding other flavors of UNIX, such as SGI's IRIX and Compaq's Tru64 Unix. The current version of WordPerfect for Windows is version 10. (*See* "Supported Platforms," http://www.corel.com/wpUnix/supportedplatforms.htm (downloaded January 15, 2002); "Product Overview," http://www3.corel.com/cgi-bin/gx.cgi/AppLogic+FTContentServer?pagename=CorelStore/ProductRequirements&id=CC1MSNDMYJC

(continued...)

what applications they want to run, corporate customers generally prefer to concentrate on a single Unix flavor (and its associated, usually proprietary, hardware). As a result, they have less flexibility in making future hardware and software choices.

200. The non-settling States' Proposal could encourage the balkanization of Windows in a pattern similar to the Unix experience, with different flavors supported by different hardware vendors. Corporate customers would no longer be free to mix Intel-based computers from different vendors (such as IBM, Compaq, Dell, and "no-name" OEMs), because some of those vendors would be likely to offer different versions of Windows combined with their own middleware and optimized for use with their own hardware.[195] IBM followed such a strategy with its OS/2 operating system, at one point offering printer drivers only for its own printers, and at another time including code designed to work best with its own proprietary video hardware.[196]

F. The Pricing Formula Poses Many Serious Conceptual and Practical Problems.

201. The non-settling States' Proposal requires not only that several versions of Windows be completely redesigned so that individual components are readily removable, but also that Microsoft give licensees discounts for each component removed.[197] I am not an economist or pricing expert, so I will not comment on the wisdom or basis in the record or law for such price regulation *per se*. As one who has been involved in many software development

(...continued)
(downloaded January 15, 2002).) Adobe sells its FrameMaker publishing program for $799 for Windows or Mac OS; the price rises to $1,329 for UNIX. (*See* "Select Products to Buy," https://store.adobe.com/WebObjects/WEC.woa/4/wo/Hl30005d200pg200AY/3.19#e (downloaded January 15, 2002).)

[195] Trial Transcript, April 9, 2002, p.m. session (Schwartz Cross), pp. 2932-2933.

[196] Charles Petzold, "The Truth about Presentation Manager," *PC Magazine*, April 11, 1989, pp. 187–203; Lenny Bailes, "OS/2: The Rewards of Patience," *PC Magazine*, November 10, 1992, pp. 233–242.

[197] Plaintiff Litigating States' First Amended Proposed Final Judgment, March 4, 2002, § 1.

projects, however, I do have an opinion as to the feasibility of implementing the pricing formula proposed.

202. I see no way that development costs can be allocated in a non-arbitrary manner to individual components, given the high degree of interdependence in both the development process and in testing. There are many joint costs in developing complex software, costs that cannot be assigned to individual components. The same people are likely to work simultaneously on multiple components, and a single component often serves multiple functions. Moreover, testing of groups of components and the system as a whole accounts for substantial costs.

203. In addition, this R&D-cost approach ignores the fact that some high-value features may have low development costs (and *vice versa*) and the fact that much of the value of the operating system derives from work done prior to the last version. For example, when I worked for IBM in the late 1960s, there were two teams working on similar timesharing operating system software products at the same time. One, TSS/360, had roughly 200 developers and failed to yield a workable product after spending an estimated $50 million and several years.[198] The other, initially named CP-67/CMS, had about 12 developers and produced what became an important IBM operating system product (VM/370) in a much shorter period of time and at a fraction of the cost. Based on the non-settling States' pricing scheme, if the first effort had yielded a minimally functional product, it would have been priced far higher

[198] I worked on the Cp-67/CMS team and know this example personally. One history of IBM's operating systems starts by stating: "In actuality, TSS never worked worth a damn" and then goes on to describe it as: "IBM knew this project was strategically important and put a lot of resources into it. Hundreds of programmers. The result was a system that was bloated and unstable. The first release took over ten minutes from power-up to the first login prompt, but usually crashed within 10 minutes after that. Eventually, IBM killed the project." (*See* "Computer History: IBM 360/370/3090/390," October 26, 2001, http://www.beagle-ears.com/lars/engineer/comphist/ibm360.htm (downloaded April 8, 2002).)

than the more valuable product that resulted from the second effort. The proposed pricing regime thus makes little sense and cannot produce a rational set of prices.

VII. Several Other Provisions in the Non-Settling States' Proposal Are Unworkable.

204. I regard the "unbinding" and the "information disclosure" provisions as the most problematic and ill-advised elements of the non-settling States' Proposal. Several other provisions, however, make little sense from my perspective as a computer scientist who has developed software and advised enterprises on their IT operations.

A. "Knowing Interference"

205. Provision 5 of the non-settling States' Proposal prohibits Microsoft from making any changes to Windows (or "Middleware") that it "knows, or reasonably should know, will directly or indirectly, interfere with or degrade the performance or compatibility of any non-Microsoft Middleware when Interoperating with any Microsoft Platform Software, other than for good cause." Although Microsoft goes to great lengths to avoid changes in Windows that "break" third-party software, including testing many applications on pre-production versions, almost any change in Windows (or any other complex software platform) is likely to affect the performance of some of the software that runs on it.[199] Under the non-settling States' definition, there is a good chance that at least a few of the programs affected would be "Middleware." Thus, one could argue, whenever Microsoft makes any non-trivial change in Windows, it "reasonably should know" that there is a high probability that at least one piece of non-Microsoft Middleware will be affected adversely. In that event, Microsoft would have to

[199] For example, in reviewing the source code for the IEXPLORE.EXE file in Windows XP, I found a block of code devoted to ensuring compatibility with Ecco Pro, a product of which I had never heard, but which I learned was a now-discontinued personal information manager (*i.e.*, calendar, address book, and task organizer) that was modestly popular before email became so ubiquitous.

determine whether it had "good cause" (which is not defined) before making *any* change to Windows.

206. Even if the "good cause" standard were clarified and set at a reasonable level, Microsoft would still suffer from the requirement to post a notice 60 days in advance of the change taking effect (*i.e.*, the new version being released). Presumably a software vendor that was affected (or thought it might be affected) would then have an opportunity to challenge Microsoft's determination that it had "good cause" for making the change. At the very least, such challenges would require Microsoft to divert technical managers from finishing the product to explaining their decisions. At worst, a negative decision could delay release of the product for many months, if not a year or more, with consequent harm to end users, third-party developers and hardware vendors that rely on timely releases of new Microsoft products.

B. "Industry Standards" Are Often Ill-Defined and Frequently Achieved Only in Part.

207. Provision 16 requires that if Microsoft says it supports an "industry standard" (broadly defined), it must support all of the standard, and that if it makes extensions, it must continue to implement the industry standard as well. In my experience, "industry standards" vary widely in their complexity and specificity. It is common for companies to support some aspects of a standard but not others. CORBA, for example, is an industry standard for communication among objects running on one or more computers. Various companies (not including Microsoft) advertise that they offer implementations of CORBA, but those implementations are generally incompatible because each supports a different subset of the "standard's" features or offers its own extensions.[200] Partial compliance and extensions to

[200] Dan Gisolfi (IBM), "Web Services Architect, Part 3: Is Web Services the Reincarnation of CORBA?" July 2001, http://www-106.ibm.com/developerworks/webservices/library/ws-arc3/ (downloaded April 9, 2002).

standards have been a fact of the computer industry from the beginning. Standards bodies, for example, started establishing specifications for computer language (such as ALGOL, COBOL, and FORTRAN) in the 1950s, but universities and vendors created many variants of these languages, each of which included extensions the creators "needed" or thought would appeal to customers.[201]

208. Determining whether a product is in full compliance with a standard is often a difficult and ambiguous process. Recall, for example, the paper I discussed earlier on the experience of integrating Microsoft's Kerberos implementation into MIT's network, which already supported two mutually incompatible versions of "standard" Kerberos. One problem MIT discovered early on was that Microsoft's implementation was not fully compliant with the standard because it had been tested using an MIT test suite later found to have bugs. Once alerted to the problem and provided with the corrected test suite, Microsoft quickly issued a fix that brought it into compliance.[202] Had this provision been in place, I presume that Microsoft could have been liable for penalties (at the very least administrative costs and delays defending against complaints from competitors) for an innocent mistake. Similarly, I have heard developers complain that to have their applications certified under some standards, they must match their program to a flawed test suite that has "bugs" that cause it to reject applications that meet the published standards.

209. My experiences with HTML are also relevant here. Although HTML has often been referred to as the "lingua franca" of the Web, in reality there are many variations and implementation differences in Web browsers as well as in their default settings. This has led me

[201] *E.g.*, one widely used version of FORTRAN was created by the University of Waterloo in Canada; it supported additional functions not in "standard" FORTRAN.

[202] Paul B. Hill, "Kerberos Interoperability Issues," LISA-NT Conference, July 30-August 2, 2000, http://www.usenix.org/events/lisa-nt2000/hill/hill.pdf (downloaded March 22, 2002).

to post notices on some of my Web sites, such as "best viewed using …." Early Web browsers often implemented the standards imperfectly. Sometimes this was done as a favor to novice HTML developers to allow their invalid HTML to be accepted. As a result, many Web sites use HTML in ways that do not match all of the written standards. In fact, these problems inspired the formation of an advocacy group, called the Web Standards Project, to encourage Web browser vendors to be more stringent in setting standards for HTML and to encourage Web authors to follow the rules more carefully. Recently this group disbanded in frustration, acknowledging that:

> Most of the Web remains a Balkanized mess of non-valid markup, unstructured documents yoked to outdated presentational hacks, and incompatible code fragments that leave many millions of Web users frustrated and disenfranchised…Browser makers are no longer the problem.[203]

210. Web browsers and other software are now implementing the HTML standards more precisely. Would Microsoft have been subject to a costly and time-consuming investigation for its earlier imperfect implementations, despite the fact that others were doing precisely the same thing and the deviation from "the standard" was designed to provide more interoperability, not less? Now that Microsoft implements the specifications more precisely, would it be subject to penalties if that more precise implementation interfered with the performance of competing Web-based "Middleware" designed (imperfectly) earlier?

VIII. The Proposed Settlement Takes a More Realistic and Measured Approach.

211. The negotiated Proposed Settlement offers several important advantages over the non-settling States' Proposal. It avoids the most serious technical problems that make the non-settling States' Proposal difficult, if not impossible, to implement. Although it requires

[203] Paul Festa, "Standards Advocates Move into Hibernation," *CNET News.com*, December 13, 2001, available at http://news.com.com/2102-1023-277029.html (downloaded January 23, 2002).

Microsoft to disclose information that ordinarily would be protected as trade secrets, it provides far more protection against the use of Microsoft's intellectual property by competitors to clone Microsoft's technologies.

A. Feasibility of Removal

212. The Proposed Settlement largely avoids the feasibility problems associated with removing "Middleware Products" under the non-settling States' Proposal.

1. *The affected "Middleware Products" would be fewer in number*, although they still would include software that is unlikely to pose the sort of middleware threat that the Court of Appeals found for Web browsers and Java. It is hard to imagine, for example, how email clients like Outlook Express could ever become significant application development platforms.[204]
2. *"Middleware Products" are defined at a less granular level*, so that Microsoft would not have to deal with the nightmare of hiding the user interfaces for individual features as opposed to larger collections of functionality.[205]
3. *Disabling end-user access, not deleting code*, is the requirement. Under the Proposed Settlement, Microsoft must make it easy for OEMs to disable end-user access to "Microsoft Middleware Products" so that OEMs can promote competing software exclusively.[206] However, the underlying code remains available for other parts of the operating system and third-party applications to use. The need for code duplication would thus be minimized and applications (existing and future) would continue to work well with whatever configurations of Windows that OEMs chose to offer.

These key differences avoid the distinct risk under the non-settling States' Proposal that Microsoft could not build a version of Windows that was both legal and workable. It also

[204] Second Revised Proposed Final Judgment, February 27, 2002, § VI.K.

[205] *See id.*, §§ VI.J, VI.K.

[206] *See id.*, § III.H.

avoids the disruption that code removal would cause to third-party application developers and the consumers who use those applications.

B. Disclosure for Interoperability, Not Cloning

213. The Proposed Settlement requires Microsoft to disclose proprietary information on various interfaces that it would normally be entitled to protect as its intellectual property.[207] However, unlike the non-settling States' Proposal, the Proposed Settlement limits the required disclosures in several ways that make it harder for competitors to misuse the information to clone features in Microsoft's products, rather than to interoperate with them:

1. *The fewer, larger pieces of "Middleware"* affected would mean that Microsoft would not have to disclose internal interfaces at the granular level required under the non-settling States' Proposal. For example, it would be obligated to disclose the interfaces used by Windows Media Player (WMP). But internal interfaces within WMP code would not have to be disclosed because WMP is one piece of "Middleware," not many, as it could be under the non-settling States' Proposal.
2. *"Interfaces" are defined in the usual way,* in terms of how other software can use them to obtain services, and would not include such details as "reference implementations" that are unnecessary for interoperability but are very valuable for cloning innovative functionality.[208]
3. *Microsoft would not be forced to share all of its intellectual property in Office and Web browsing*. There is no open-source requirement for IE, the express purpose of which in the non-settling States' Proposal is to let competitors have free access to Microsoft's intellectual property. There is also no forced auction of the vast amounts of intellectual property embodied in Microsoft Office (and parts of Windows on which it depends for some functionality).

[207] *See id.*, § III.D.

[208] *See id.*, § III.D.

4. *Exceptions for security are included in the Proposed Settlement*. These exceptions would under certain conditions allow Microsoft to keep confidential information that could be misused to compromise the security of installations.[209] Although various of the non-settling States' witnesses have sought to portray these provisions as "loopholes,"[210] some restrictions are needed and I would be surprised if Microsoft were able to misuse these provisions under the detailed scrutiny it will continue to face.

C. Other Provisions Show Similarly Sensible Limits.

214. At a broader level, the Proposed Settlement shows many signs of having been carefully negotiated with technically knowledgeable input, rather than being a compilation of competitors' wish lists. For example, it does not impose a vague and unworkable requirement that Microsoft make changes in its products only for "good cause." But Provision III.F.1 of the Proposed Settlement would prohibit Microsoft from retaliating against any hardware or software vendor for "developing, using, distributing, promoting or supporting any software that competes with Microsoft Platform Software or any software that runs on any software that competes with Microsoft Platform Software."[211] This provision provides a vehicle for challenging changes in Windows designed to harm competitors, without stalling the development of Windows or forcing Microsoft to document a "good cause" for every change in the system.

215. The Proposed Settlement also sensibly stays away from regulating how Microsoft complies with "Industry Standards." Professional developers and IT managers in enterprises are generally a technically savvy group. They are used to evaluating the reality

---

[209] *See id.*, § III.J.1.

[210] *See*, *e.g.*, Direct Testimony of Dr. Andrew W. Appel, March 13, 2002, ¶¶ 8, 116, 126; Direct Testimony of James Barksdale, March 19, 2002, ¶¶ 135-136; Direct Testimony of Dr. Carl S. Ledbetter, March 27, 2002, ¶ 176; Direct Testimony of Steven McGeady, March 26, 2002, ¶¶ 56, 69; Direct Testimony of David Richards, March 20, 2002, ¶ 99; Direct Testimony of Jonathan Schwartz, March 28, 2002, ¶ 171; Direct Testimony of Michael Tiemann, March 21, 2002, ¶ 185.

[211] Second Revised Proposed Final Judgment, February 27, 2002, § III.F.1.

behind vendors' claims that they support this or that standard, or provide this or that functionality. They also know that even if two vendors support a standard, there is no guarantee that their implementations will work together seamlessly. It would make little sense for the court (or a court-appointed special master) to put itself in the position of trying to police adherence to industry standards.

D. Conclusions

216. Implementing the provisions of the non-settling States' Proposal would greatly slow, if not end, innovation in Microsoft's major products, its Windows operating systems, and its Office suite. Indeed, there is a substantial probability that Microsoft would be unable to develop and market a workable version of Windows that complied with the requirements. In that case, Microsoft would have to continue to apply for extensions to market the existing versions. Even if the provisions proved technically feasible, they would at a minimum redirect Microsoft's development resources from improving the operating system to making it compliant with the unbinding requirement and to providing competitors with the information demanded by the disclosure provisions.

217. If many licensees took advantage of the regulated discounts required under the non-settling States' Proposal by removing significant amounts of "Middleware" code in Windows, the value of Windows to consumers and developers would also be sharply diminished. End users would have to replace existing applications or find ways to replace the Windows elements deleted by their suppliers. Developers would have to rewrite their applications to account for a diverse and inconsistent universe of versions of Windows.

218. Implementation of the non-settling States' Proposal also would strip Microsoft of much of the value of its intellectual property. In the case of IE and Office, the transfer of intellectual property is the explicit goal. In the case of Provision 4, the loss of intellectual

property would be implicit, done in the name of "interoperability." But that provision (in combination with other elements, including the definition of "Middleware") goes far beyond what developers need to achieve interoperability. The information required under that provision would do little to further interoperability, but it would make it a great deal easier for competitors to clone features in Windows.

219. The beneficiaries of implementation of the non-settling States' Proposal would be the same companies that are its most vocal advocates, Microsoft's competitors. Their most immediate advantage would come from degrading the performance of Windows. Over the longer run, they would benefit from halting or slowing the pace of innovation in Windows and from the ability to obtain free access to much of Microsoft's intellectual property. Armed with such information, they would be well positioned to imitate whatever features in Windows they wished.

I declare under penalty of perjury that the foregoing is true and correct. Executed this 25th day of April, 2002, at Cambridge Massachusetts.

_____

Stuart E. Madnick