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 BILL GATES

SUMMARY TABLE OF CONTENTS

TABLE OF CONTENTS

DIRECT TESTIMONY OF BILL GATES

1. My name is William H. Gates III. I am the Chairman of the Board and Chief Software Architect of Microsoft Corporation. I am a co-founder of Microsoft and was its Chief Executive Officer from 1986, when the company went public, until January 2000. I made the decision to relinquish my duties as CEO so that I could spend more time working with the product groups at Microsoft to develop new technologies that will be necessary to enable the next generation of computing.

2. My work at Microsoft has encompassed nearly every aspect of the software business. I have written code and architected software products. In my positions at Microsoft I have met regularly with the leaders of small and large companies in the personal computer industry and in related businesses, including companies Microsoft works with to develop and distribute products, as well as with customers in the business world, in academia and in government. I closely track trends that might affect Microsoft's business.

3. I am married and have two children. I have written two books, "The Road Ahead" (1995) and "Business@ the Speed of Thought" (1999), both of which deal with how innovative software technology can increase business productivity.

4. I am submitting this testimony to provide the Court with information concerning Microsoft's development of new platform technology and how that technology is used by computer manufacturers, software developers and others in building new products. I believe that Microsoft has made a significant contribution to the success of the computer industry through our development and broad licensing of our Windows family of operating systems and other software products.

5. In my testimony, I identify a number of key aspects of Microsoft's business and technology model that have been vitally important to the value that Windows provides to the marketplace and thus to its success. The non-settling States' proposed remedies ("NSPR") would imperil Microsoft's business and technology model, depriving the marketplace of the primary benefits that Windows provides and thereby greatly devaluing the product.

6. In fact, for reasons that I explain in this testimony, I believe that Microsoft would be unable to develop a version of Windows that would comply with Section 1 of the NSPR as written. I understand that if that were the case, Section 1 would ban Microsoft from continuing to offer Windows in the marketplace, unless the Court later agreed to modify the remedy.

7. Section I of my testimony describes Microsoft's formation and our early development of operating system products. I explain how Microsoft's creation and nurturing of a common platform that ran across competing brands of personal computers ("PCs") helped to unify a fragmented industry by providing consumers with the benefits of broad interoperability across PCs from many computer manufacturers ("OEMs") and applications from many independent software developers ("ISVs"). In so doing, Microsoft helped to bring about a revolution in computing. I also briefly describe Microsoft's efforts via its .NET and Trustworthy Computing initiatives to develop breakthrough technologies that, if successful, should spark a new round of investment and excitement in computing, providing business opportunities for thousands of companies and benefits for millions of consumers.

8. Section II addresses three key aspects of Microsoft's business that have been vitally important to its success and the success of all those who build products that take advantage of the Windows platform. Those elements are: (i) the stability, consistency and quality of Windows and Microsoft's commitment to developing innovative new releases of the operating system that provide ISVs and consumers with new capabilities; (ii) the interoperability across a wide range of hardware and software provided by Windows and associated tools and documentation; and (iii) the incentive to innovate provided by the promise of intellectual property protection. The NSRP would undermine all three elements of Microsoft's success, causing great damage to Microsoft, other companies that build upon Microsoft's products, and the businesses and consumers that use PC software.

9. Section III provides a section by section analysis of the problems inherent in the NSPR, covering all the substantive provisions of the proposed remedy. The NSPR would deprive Microsoft of much of the economic value of its two most important products, Windows and Office, effecting a massive transfer of Microsoft's intellectual property rights in both products to its competitors. With free access to Microsoft's technology, competitors could build a clone of Windows (a product that mimics the key features of the operating system) and their own version of Office without bearing any significant part of the R&D expenses that Microsoft incurred to build the technology. As I understand it, providing Microsoft's technology to its competitors so they can build "functional equivalents" of our products now, and match all our future innovations for ten years, is in fact one of the central objectives of the NSPR.

10. As explained in Section III, the NSPR would undermine the Windows platform, to the detriment of all who benefit from it, in many different ways. In fact, the NSPR would hobble Microsoft as a competitor and innovator across many product categories because many of its provisions are broadly worded to apply to *any* Microsoft product, service, feature or technology.

11. Aside from these concerns, it would be extremely difficult, if not impossible in some cases, for Microsoft to comply with the NSPR. Many key aspects of the NSPR, particularly its definitions relating to "middleware," are vague and ambiguous, providing Microsoft with no clear statement of its obligations. Other aspects of the NSPR simply could not be feasibly implemented. Many provisions of the NSPR lead to extreme results, but Microsoft would not have the freedom to construe the NSPR in ways that we find less extreme. Microsoft is committed to complying fully with Court orders, including any remedy that may be ordered in this case. We can do that only if the remedy is clear as written and its terms feasible.

I. MICROSOFT'S ROLE IN FOSTERING INNOVATION

12. For more than 25 years, Microsoft has been at the forefront of the development of the PC industry. Microsoft has played an important role by creating software—MS-DOS and later Windows—that created substantial business opportunities for other companies, which in turn made our operating systems more valuable. Today Microsoft is investing in a next generation computing platform, XML Web Services, that holds the potential to unleash new waves of productivity gains in the economy.

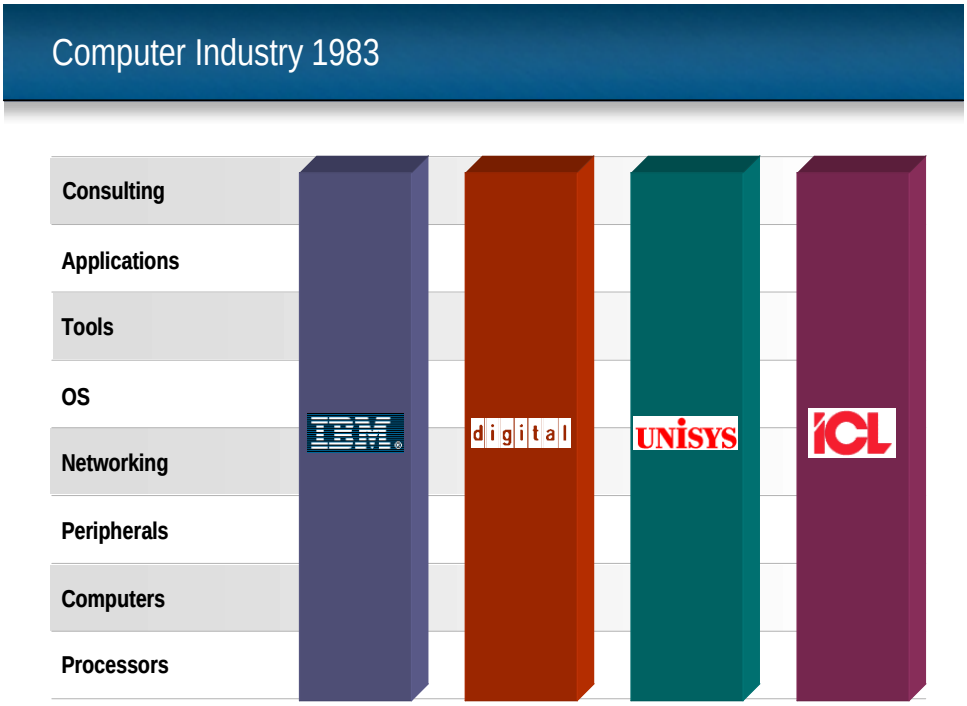
A. MICROSOFT'S DEVELOPMENT OF A STANDARD COMPUTING PLATFORM

13. In the 1960s and 1970s, IBM dominated computing. IBM made "mainframes," large computers such as the System 370 that performed data-intensive functions for large corporations. These systems were expensive to acquire (often $1,000,000 or more) and expensive to operate. Maintained by technicians in air conditioned rooms behind glass walls, few had access to the computing resources that mainframes provided.

14. Typically these mainframes were provided to corporate customers with all the software necessary to make the mainframe useful. Software was not considered to be a distinct line of business. In those days there was no such thing as a "software company."

15. Apart from IBM, mainframes were offered by competitors such as Data General, Sperry-Rand, Burroughs and others. There was little or no interoperability among mainframes from each company. A corporate customer would choose an "all IBM" solution or an all "Burroughs" solution, etc., for a particular computing need.

16. In the 1970s, Digital Equipment Corporation achieved considerable success with a line of less expensive computers (called minicomputers) that were particularly well suited to engineering and scientific tasks. Again, however, there was little or no interoperability between DEC minicomputers and mainframes offered by IBM and others. And these computers were still far too expensive to be within the reach of most consumers or small businesses. The largely "vertical" nature of the industry at this time is depicted below.



17. In 1975, my friend Paul Allen and I saw an article in a magazine called *Popular Electronics* describing a new computing device called the MITS Altair 8800. This device was a far cry from what people think of today as a personal computer. Ordered through the mail, the Altair arrived as a box of parts with some instructions on how to assemble it. As such, the Altair was strictly for computer hobbyists. When assembled, the Altair was a breadbox-sized machine with rows of switches and a few lights that, with the addition of some software, could be made to blink.

18. Although the Altair did not do much, Paul and I recognized that it was the start of a trend that had the potential to revolutionize the computer world. The Altair was made possible by a new invention: a computer on a single integrated circuit (or "microprocessor"). The microprocessor integrated circuits enabling thousands (and today, many millions) of functions. The advent of the microprocessor carried with it the possibility of making computing power far more affordable and thus widely available.

19. We recognized that to make the Altair and other microprocessor-based devices useful, they were going to need software. I left college, and Paul and I founded a company to develop great *micro*processor *soft*ware, which we called Microsoft. It was not much of a business in its early years, just Paul, me and a small group of developers we hired banging out code day and night in spartan offices in Albuquerque, New Mexico. But it was a labor of love.

20. Our primary product, Microsoft Basic, was a programming language, the first for this new breed of computing device. During our first five years in business, our products were directed almost entirely at helping software developers create applications for this device. From the start, we recognized that the potential of microprocessor-based computers (or microcomputers, as they became known) would not be realized unless a broad array of useful software products were written to run on them. Although the market at the time was tiny, we aimed to fill the need for programming languages and software tools to make it easy to develop microcomputer software. We knew that our vision could not be achieved without the creativity and innovation of many developers, including developers working for other companies. So we began Microsoft with a unique focus: building tools to enable thousands of software developers to create the vast range of applications that would ultimately make computers useful for consumers and businesses of all sizes.

21. By 1980, a number of companies were offering microcomputers that were the precursors to today's personal computer. Unfortunately, however, the personal computer industry was developing along the lines of mainframe computing: the offerings of various computer manufacturers were separate islands in the sea of computing. Early

PCs such as the Tandy TRS-80, the Apple II, the Commodore PET, the Atari 800 and others each ran their own, distinct operating systems. That meant that software applications written for one line of computer would not run on any other line. End users thus could not easily share information among computers and they had to learn a different set of skills to operate each one. In other words, the fledgling PC industry was fragmented.

22. In mid-1980, IBM approached Microsoft with its plans to introduce an IBM "personal computer" (a term that IBM coined). We were very enthusiastic about this development. We hoped that the resources, expertise and cachet of IBM would help people to recognize that the PC could be much more than a "toy," as many then regarded it to be.

23. IBM was on a fast-track to develop its PC, and asked if Microsoft (by then based near Seattle) would license our BASIC language product in order to make their PC more accessible to software developers, which we did. They also asked that we supply operating system software for the device. We agreed to do so. We quickly acquired rights to a small operating system program (called Q-DOS, for Quick and Dirty Operating System) and hired its developer, Tim Paterson, to work with us to develop an operating system that would meet IBM's needs. Within nine months we produced an operating system that met IBM's specifications. Since that small beginning, Microsoft has invested more than $6 billion in developing its operating system products (and billions more in creating development tools and applications that enhance the value of the platform).

24. When released in 1981, the IBM PC was available with a choice of three operating systems: CP/M-86, UCSD-P System and MS-DOS from Microsoft. Over time Microsoft's operating system became the most popular because we worked

relentlessly to improve it, adding new features and other innovations to the platform, we widely licensed it at attractive prices as a consistent software platform that would promote interoperability among a wide range of hardware and software products, and we helped developers to build applications for the operating system.

25. Early on, we recognized that consumers would benefit greatly if a wide range of hardware and software products could interoperate with one another. Among other things, (i) the products would be more useful if information could be exchanged among them, and (ii) development costs would fall and a broader array of products would become available if they could be developed for larger customer segments without the need to rewrite software to target narrow platforms. As more products became available and more information could be exchanged, more consumers would be attracted to the platform, which would in turn attract more investment in product development for the platform. Economists call this a "network effect," but at the time we called it the "positive feedback loop."

26. Given these benefits, we expected that the market would attach great value to any product that enabled such broad interoperability. As I explain more fully below in Section II.B, Microsoft committed itself to providing compatibility among a wide range of products, as we believed the market would demand. There were three key and closely-interrelated elements to our strategy, a strategy that is unchanged to this day.

27. *First*, we worked hard to develop MS-DOS (and later Microsoft Windows) as a useful platform for software developers. We built capabilities into MS-DOS which developers could draw upon in building their own applications, freeing them from the need to recreate these capabilities on their own. With each succeeding release of MS-

DOS, we added new capabilities (all of which consumers take for granted today), thereby facilitating the efficient development of innovative new applications. Then, as now, these capabilities are "exposed" to developers via application programming interfaces or "APIs." As we added new APIs, we also went to great lengths to continue to support existing APIs so that older applications would run well on newer versions of MS-DOS and Windows. We thus preserved the integrity and utility of our operating systems even as we drove the platform forward with new technologies such as the graphical user interface in the late 1980s and Internet technologies such as HTML in the mid-1990s.

28. *Second*, we broadly licensed MS-DOS at low prices to any OEM that was interested. This freed developers from the need to create entirely separate applications for each line of PC. With very minor exceptions not relevant here, MS-DOS was the same on each line of PC from different OEMs, masking differences in the underlying hardware platform. As a result, developers knew that they could create an application to run on MS-DOS and that it would run not only on the IBM PC, but also on any other PC that used MS-DOS as its operating system. OEMs, in turn, knew that they could build a PC based on MS-DOS without bearing the substantial costs of designing, developing and testing an operating system themselves, and that their PC then would run the growing body of applications written to MS-DOS, making their PCs more valuable to consumers.

29. *Third*, we actively assisted software developers in making use of the APIs exposed by MS-DOS, providing them with software development tools and technical information concerning the operating system, and instructions. We called the employees who spread the word of the platform benefits of MS-DOS "evangelists," reflecting the passion and commitment with which they went about their jobs. Today Microsoft invests

many hundreds of millions of dollars annually in developing tools, information and other resources to facilitate the development of products that interoperate with our operating system. Our Developer and Platform Evangelism Division, some 2,500 employees strong, is dedicated to this work. Hundreds of evangelists are stationed around the world to assist developers in building products that interoperate with Windows.

30. Microsoft's development and evangelization of MS-DOS (and later Windows) was critical to the creation of the PC industry. At first dozens, later hundreds, and eventually thousands of OEMs came into existence and built PCs based on the R&D work of Intel, AMD and others on compatible microprocessors and Microsoft on successive versions of MS-DOS. Similarly, thousands of ISVs were formed to build software applications that took advantage of the ever-growing range of capabilities provided by MS-DOS, which in turn was made possible by exponential growth in the underlying processing capability of microprocessors.

31. With the success of MS-DOS and the products built to work well with it, the PC industry was organized along a "horizontal" model, quite unlike the model that prevailed in the days of mainframe computing or even in the early days of the PC industry. Under this model, shown below, a wide variety of companies provide hardware and software products and services that are broadly compatible with one another.

32. In view of the close interrelationships among PC hardware and software products, the industry is often referred to as the "*PC ecosystem*." Microsoft's Windows operating system is a key component of the PC ecosystem, and thus the health of the ecosystem depends in substantial part upon the continued health of and improvements to Windows.

33. The PC industry has been enormously successful. This year the PC industry will generate hundreds of billions of dollars in revenue to a great number of companies, providing the marketplace with a broad range of interoperable products, from PCs to motherboards, video and audio cards, printers, monitors, software of all kinds, and much more, delivered through a wide variety of sales channels and supporting a variety of closely related businesses such as training centers, value added retailers, and so forth. Although the industry has recently experienced a cyclical downturn, I am confident that with continued innovations in operating systems and other key technologies, PCs will

become much easier to use and more useful, sparking new rounds of productivity gains and empowering users.

34. Many of Microsoft's most important competitors have chosen alternative business and technology strategies, often focused more on selling relatively expensive hardware than on facilitating broad interoperability. For example, Apple has historically chosen not to license its Apple Macintosh operating system products to other OEMs, preferring to obtain the competitive advantage it perceives from being the only OEM to offer PCs that feature the Mac OS. While offering Apple certain advantages (such as the ability to ensure that its hardware and software are optimized to work very well together), that strategy provides consumers with much less choice in software and hardware that work with the Mac OS.

35. In mid-1985, I urged Apple to license its Macintosh operating system software to other OEMs, and I offered to help Apple do so. (My July 8, 1985 letter to John Sculley, and an accompanying memorandum dated June 25, 1985 setting out my views more fully, is in evidence as DX 2245 and is submitted here as DX 1558. A follow-up memorandum that I sent to Mr. Sculley on July 29, 1985 is in evidence as DX 2246 and is submitted here as DX 1559.) Among other things, I explained to Apple that I was enthusiastic about the benefits of licensing Mac technology to other OEMs because "[t]he IBM [PC] architecture, when compared to the Macintosh, probably has more than 100 times the engineering resources applied to it when investment of compatible manufacturers is included." DX 1558. (The term "IBM architecture" referred generally to PCs that were built in accordance with IBM's original decision to use an Intel chip and Microsoft operating system.) Microsoft was interested in promoting the development of the Apple

Macintosh platform because Microsoft was the leading ISV for Apple and enthusiastic about the capabilities of the Apple platform.

36. Today one of Microsoft's most important competitors is Sun Microsystems. Sun is primarily focused on selling network server hardware. Like Apple, Sun's software efforts are primarily directed at promoting its hardware business. Sun also offers its own, proprietary version of UNIX called "Solaris." Solaris is mostly used to run on hardware offered by Sun itself. UNIX applications must be customized, at considerable expense, to take full advantage of the Solaris operating system.

37. I believe that Sun is feeling considerable competitive pressure from OEMs such as Compaq, Dell and Hewlett-Packard that are building server hardware that use server versions of Windows operating systems. In its server operating system business, Microsoft is following the same high volume/low cost strategy that has been so successful on the desktop. By making its server operating systems widely available to OEMs at attractive prices, Microsoft enables OEMs to build server hardware that is substantially less expensive than Sun's offerings.

B. New Technology Initiatives

38. Microsoft is not content to rest on the extent to which software presently helps people as they go about their daily lives. While PCs are used by nearly every knowledge worker and in more than half of American homes (a success rate that seemed barely imaginable twenty years ago), we believe that breakthrough technologies now under development can make software vastly more useful, easier to use and more reliable. Our strategic direction in technology development is called ".NET." .NET

includes our work on Trustworthy Computing, an initiative to help bring about computers that are far more available, reliable and secure than they are today.

1. .NET

39. The broad connectivity among computers provided by the Internet provides an infrastructure that software developers can use to build new products that will greatly promote interoperability among computers running any operating system, from mainframes to PCs to small, handheld devices. Microsoft is at the forefront of industry efforts to define new technical standards, promulgated by standards bodies such as the World Wide Web Consortium, that will facilitate the exchange of information and functionality across widely disparate computing platforms.

40. These emerging standards collectively define technology known within the industry as "XML Web Services." XML Web Services can be thought of as software programs that provide functionality to other programs, located on the same computer or other computers (around the corner or around the world), through a defined set of industry-standard interfaces. Of particular note here, the interoperating computer programs can be run on any operating system. Early examples of XML Web Services are already up and running on Windows and various UNIX operating systems.

41. Microsoft was an early pioneer in defining and standardizing XML and associated technologies. Today Microsoft is working closely with IBM and others to define and enhance standards relating to XML Web services, working through the World Wide Web Consortium, the new Web Services Interoperability Organization, and other standards bodies.

42. Microsoft's implementation of XML Web Services standards is an important part of our .NET efforts. Our goal is to deliver the best implementation of a platform for building XML Web Services. As a result, we will strive to provide the best price/performance, the best scalability across multiple computers and the best development tools to take advantage of the platform. We believe that we are well situated to deliver on our .NET platform vision because of our deep experience in developing and evangelizing platforms that have a proven track record of meeting industry needs.

43. Our key developer tool for building next-generation software, Visual Studio.NET, was released earlier this year following more than three years of development work. It has been very well received by the developer community. Using Visual Studio.NET, developers can create software products in any of approximately 30 programming languages, including Java, that will run on Microsoft's .NET platform software. (Most programs written for Sun's Java platform are written in a single programming language, Sun's own Java language.)

44. Microsoft also developed a brand new programming language, called C# (pronounced "C-sharp"), that is tuned to the needs of developing XML Web Services. Microsoft submitted the specifications for C# to the European Computer Manufacturers' Association (ECMA) so that the language could become an industry standard, and ECMA adopted C# as a standard in December 2001. (By contrast, Sun has touted Java as an industry standard, but withdrew it from the ECMA standard setting process and maintains proprietary control over the language.)

45. Software products written using Visual Studio.NET will interoperate with products that adhere to the relevant industry standards for XML Web Services *whether*

or not those programs are running on any Microsoft operating system or other Microsoft platform software. That is the beauty of XML Web Services—they enable access to data and functionality any time, on any device, running any software.

46. Realization of our .NET vision will take many years and many billions of dollars of R&D investment by Microsoft and partners who share our vision. If successful, however, we believe that .NET will bring about growth in economic productivity that will exceed the productivity benefits to date from the development and broad adoption of PC technology.

2. Trustworthy Computing

47. To make computing more pervasive, the industry needs to build systems that excel in availability, reliability and security, which I call "Trustworthy Computing." Today the PC ecosystem falls short in all three respects. Internet connections may fail, software programs may crash, and viruses may infect any computer that interacts with any other. Even when working as designed, computers remain too hard to use. Absent sustained effort to attack these engineering challenges, the problem is likely to get worse, not better, as the computing environment becomes ever more complex with greater interoperation of a broad range of devices via the Internet and other networks.

48. I have committed Microsoft to developing software that will promote Trustworthy Computing. Our Visual Studio.NET development tool is an important step in this direction. Visual Studio.NET is the first multi-language development tool that is optimized for the creation of secure code. Advanced programming techniques embodied in Visual Studio.NET will also enable developers to write "managed" code—code that is less prone to a wide variety of programming errors.

II. KEY ASPECTS OF MICROSOFT'S BUSINESS THAT ARE IMPERILED BY THE NSPR

49. Three overarching aspects of Microsoft's business and technology model are imperiled by the NSPR: (i) the considerable benefits provided by Microsoft's ongoing development of successive versions of Windows as consistent, well-tested, well-supported platforms for software development; (ii) Microsoft's efforts to promote the development of a broad range of hardware and software products that interoperate well with one another; and (iii) the central role played by intellectual property protection in providing an incentive for Microsoft to invest capital, time and energy in software development. I discuss each of these in turn.

A. THE PLATFORM VALUE PROVIDED BY WINDOWS

50. The primary value that Windows provides to the economy is that it is a full-featured platform for software development that promotes compatibility across a broad range of hardware and software products. I believe that many provisions of the NSPR would prevent Windows from continuing to provide that value.

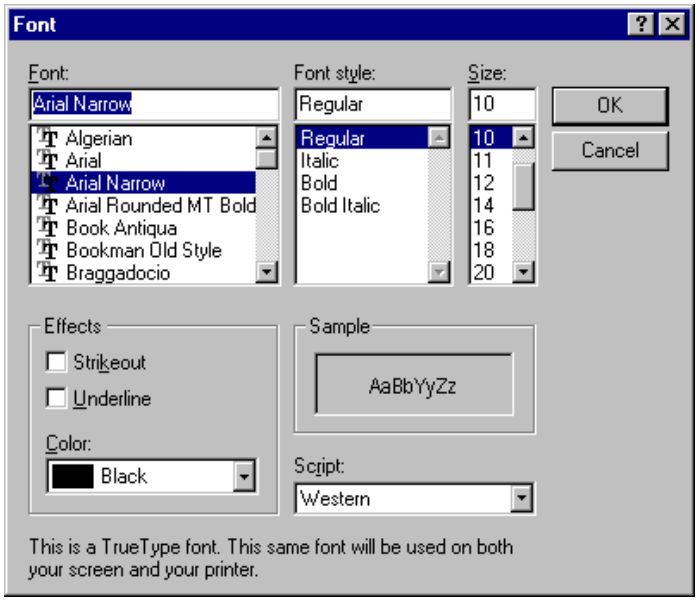
51. In the discussion that follows in this subsection, I explain why it is highly efficient for software programs to be built on high-quality software platforms. I then identify three aspects of Microsoft's development of Windows that are essential to the success of that product, all of which are imperiled by the NSPR's vision of a world in which OEMs and third parties could remove important software code from Windows as well as by other provisions of the NSPR.

52. A "platform" is software that provides capabilities that can be used by ISVs in creating their own software programs. Modern computer operating systems usually serve as platforms.

53. Developers are free to use as few or as many of the capabilities in an underlying platform as they like. Developers are always free to create their own programs to provide any capability provided by Windows (and to draw upon other Windows capabilities in doing so).

54. Capabilities provided by operating systems are often referred to as "system services." A software program can make use of a system service by making a "call" to an API. When an application "calls" an API, the operating system performs the function associated with that API by causing the underlying microprocessor to perform a specified set of instructions. For example, an ISV can call upon Windows to cause a set of user interface controls, such as toolbars, menus, or back and forward buttons, to appear in the ISV's application.

55. A simple example of an API is the ChooseFont API. By calling this Windows API, an ISV is able to display the following dialog box in its software program.



By calling a set of Windows APIs that relate to fonts, an ISV can create a software program that easily enables users to create documents using a wide variety of fonts provided by Windows.

56. Windows exposes a very rich set of capabilities for ISVs to use today through more than 6,000 APIs (in very round numbers). These APIs are generally referred to in the industry as the "Win32 API set" because the APIs were introduced in connection with Microsoft's 32-bit operating systems. We have added to and improved upon these APIs as our operating system technology has evolved.

57. The availability of a consistent, high quality software platform frees ISVs from the need to repeatedly "reinvent the wheel." Rather than expend resources to develop technology already developed by others, ISVs can simply call on the underlying operating system, often with just a few keystrokes. That frees ISVs to focus on building the unique features of their own products and very substantially reduces development time and costs.

58. The availability of consistent, high quality platform software also promotes compatibility among software programs from a wide range of developers, which in turn makes software programs easier to develop and use. For example, the Internet Explorer system services in Windows provide an implementation of HTTP, a key industry-standard protocol for transmitting information (such as Web pages) over the Internet. By calling on the HTTP support in Windows, any ISV can develop an application that can upload and download information to and from other programs in a compatible way (even if those programs are running on a UNIX operating system, as they often are on the Web),

avoiding various problems that may arise if each ISV developed its own implementation of HTTP.

59. Another example of a system service in Windows that promotes interoperability is Microsoft's Component Object Model or "COM." COM provides a set of methods, exposed via Windows APIs, for ISVs to exchange information and functionality among software programs. When you embed an Excel spreadsheet in a Word document, for example, you are using COM services provided by the underlying Windows operating system. Thousands of software applications take advantage of the COM technologies in Windows.

60. The availability of a stable, consistent set of system services in Windows also enables ISVs to create programs that adhere to common user interface elements, promoting compatibility among programs. For example, consumers who become familiar with the "ChooseFont" dialog box shown above when working with one program will feel comfortable when presented with the same or a similar dialog box in other Windows-based programs that call upon it. Similarly, Windows exposes APIs that make it easy to create the "toolbars" that often appear at the top of applications. End users, who learn how to use icons, drop down menus and the like in one application written for Windows, will likely have an easier time learning how to use similar features in other Windows-based applications, particularly if the ISV elects to conform to user interface conventions established by Microsoft to promote such ease of use. (ISVs who wish to develop their own font controls or toolbars are of course free to do that.)

61. I believe that through years of effort and billions of dollars in R&D, Microsoft has done a better job than any other company at providing ISVs and the PC

industry generally with the benefits of a platform for software development. Three key aspects of our work in Windows, however, are imperiled by the NSPR. These three aspects are discussed below.

1. The Integrity of the Windows Platform

62. The NSPR would undermine the utility of Windows as a development platform by requiring Microsoft to enable anyone who offers to license 10,000 copies to remove blocks of software code from Windows before providing it to consumers. (Sections 1, 2(c), 7 and 22.x.) If software code is removed, the Windows APIs provided by that code will no longer function. That means that applications that call upon those APIs will fail to function properly or may not run at all (depending upon which APIs are rendered inoperable).

63. If OEMs and others remove software code from Windows, as the NSPR would authorize them to do, then the Windows platform would "fragment." Once fragmented, Windows no longer would provide ISVs with a well-defined set of APIs on which they can rely to provide useful functionality. Under the NSPR, one OEM might ship software marketed as "Windows" without its Web browsing software, another OEM might ship software marketed as "Windows" without its media creation, delivery and playback software, while a third OEM might ship software marketed as "Windows" without its instant messaging software, all categories of software that provide useful APIs to ISVs. Other OEMs might remove smaller parts of Windows, such as its HTML rendering engine (a part of the Web browsing software in Windows) or even subsets of that.

64. Sections 1 and 22.x.i of the NSRP identify categories of software that Microsoft must make optionally removable within six months of entry of the NSPR, ten of

which cover software in existing versions of Windows. Additional software in other unspecified categories also must be made optionally removable under Section 22.x.ii by an unspecified date. Assuming that just ten blocks of code were made optionally removable under the NSRP, however, there would be more than 1,000 variations of Windows that OEMs and others could create from each release of Windows.

65. If OEMs installed idiosyncratic variations of Windows on new PCs, ISVs seeking out to build new applications could no longer rely on Windows to provide useful functionality. On any given installation of Windows, useful system services might be available, or they might not, depending on which software code the OEM elected to remove. ISVs are less likely to call on services provided by Windows APIs if their customers may be running variations of Windows from which APIs have been removed. ISVs would then be required to develop and test customized versions of their applications for each major variation of Windows.

66. Customers would soon be faced with the prospect of finding and distinguishing among, for example, Corel WordPerfect for Compaq Windows, Corel WordPerfect for Dell Windows and Corel WordPerfect for Gateway Windows (and for Sun Windows and AOL Windows), each with varying capabilities reflecting the underlying capabilities of the version of Windows to which they were written. Software innovation would slow as ISVs devoted greater resources to (i) duplicating functionality that Windows might otherwise provide and (ii) testing many variations of their products to reflect variations in the underlying operating systems.

67. Once multiple versions of popular applications were created to run on specific versions of Windows, OEMs could no longer provide Windows-based PCs to

customers with the value proposition that customers could run any of thousands of Windows applications on their PCs. Far from providing any assurance of quality, the Windows brand would mean whatever anyone who offers to license 10,000 copies wanted it to mean (or whatever their sub-licensees wanted it to mean).

68. Consumers expect to know how to use their home PC after learning how to use a PC at work or at school, and vice versa. Once various idiosyncratic versions of Windows are created, however, consumers would lose the ability to easily transfer their learning from one PC to another.

69. In short, *if the Windows platform were to fragment, the primary value it provides—the ability to provide compatibility across a wide range of software and hardware—would be lost.* Windows would no longer offer an efficient platform to ISVs because Windows would not consist of any single platform on which ISVs could rely in developing applications. (*See* Demonstrative Exhibit 1.)

70. As software programs became more costly to develop and offered fewer new innovations, consumers would have less incentive to buy new PCs. The same "positive feedback loop" that propelled the PC industry to years' of steady growth would work in reverse, causing the industry to stagnate as products became more expensive to develop even as they provided fewer benefits and less interoperability.

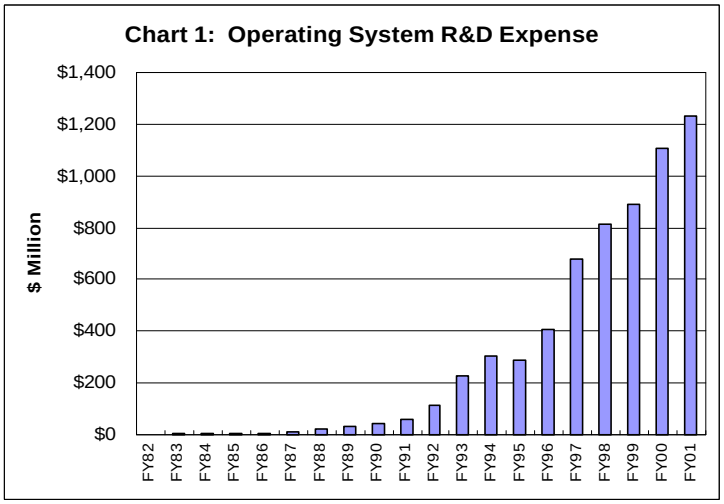
2. Innovation in Windows

71. One of the reasons Microsoft's operating system has been successful is that ISVs and OEMs know that Microsoft is committed to working tirelessly to improve the Windows platform by providing new capabilities made possible by more powerful microprocessors. As famously stated in "Moore's Law," microprocessor power has grown

exponentially, doubling roughly every 18 to 24 months. Today's microprocessors provide more computing power than an IBM mainframe of a generation ago. Rapid growth in microprocessor power makes possible major new innovations in the next layer of the computing architecture – operating systems. New capabilities in operating systems, in turn, enable ISVs to provide new innovations in their applications. The availability of new applications spurs sales of new PCs.

72. Absent steady advances in operating systems and applications, consumers have little reason to buy new software products since software never wears out. That is why it is especially important in the software industry that new product releases provide consumers with new capabilities. The point is well-illustrated by Microsoft's development of Windows 95, an operating system that provided the industry with a wide range of advances, sparking years of very strong growth in PC hardware and software sales.

73. Microsoft's large and rapidly growing investments in operating system R&D reflect our belief that improving Windows, in part by adding new capabilities, is vitally important. As Chart 1 shows, we are committing ever greater resources to improving our operating system products (both client and server), which provides direct benefits to consumers.



74. By providing OEMs and others with the right to remove software code in at least ten categories from Windows, the NSPR would essentially "draw a line

around Windows," as competitors once urged the DOJ to do. If OEMs and others were free to remove software code from the ten or more so-called "middleware" categories from Windows, all that ISVs could rely upon in developing new products would be the relatively small and ill-defined subset of Windows that remained. I believe that some of the witnesses for the non-settling States have referred to this as the "core Windows operating system," although the NSPR provides no statement as to what would constitute the "core."

75. Similarly, it appears that Windows itself could not rely upon any software code that fell into any of the ten or more "middleware" categories. This point is unclear because Section 1 of the NSPR provides no clear guidance and the testimony of the non-settling States' witnesses are in conflict with one another. That means that Section 1 would expose Microsoft to contempt liability for engaging in the basic engineering practice of pursuing an integrated design, *i.e.*, a design whereby one part of Windows would rely on capabilities provided by another part of Windows.

76. For example, Microsoft developed a new "Help" system in the mid-1990s that presented information about how to use Windows in the HTML format, thereby enabling a richer presentation of data, use of Back and Forward buttons and hyperlinks (as used in a Web browser), and other advantages. As is Microsoft's usual practice, we also exposed the capabilities of our new HTML-based Help system to ISVs through published APIs. These APIs enabled ISVs quickly and easily to create HTML-based Help systems for their own applications. The Help systems in future versions of Windows will provide Help through short video clips with audio, which many people find easier to follow than text. Some of our applications already do so today.

77. The NSPR may prevent Microsoft from providing these new capabilities in Windows. Our software for displaying information in the HTML format is provided by the Web browsing software in Windows, a category of software that Microsoft must enable OEMs and others to remove. Section 1 states that despite the removal of such software code, the operating system must "perform effectively and without degradation (other than the elimination of the functionalities of any removed Microsoft Middleware Product)." Section 2(c) states flatly that any OEM and others can "remove the code for Microsoft Middleware Products."

78. I cannot tell from these sections and the associated definitions what software files (or routines within files) Microsoft must find a way to make "removable" from Windows and what features must still "perform[ ] effectively and without degradation." For instance, I understand that the non-settling States' Rule 30(b)(6) witness, California Assistant Attorney General Tom Greene, testified that it would be permissible if the removal of "middleware" broke "audio" Help, but not text-based Help. As I understand it, Mr. Greene testified that text-based Help is a "basic" operating system function, and thus should remain no matter what software code is removed, but that "audio" Help is not. (Deposition of Tom Greene, March 12, 2002 at 66-67.)

79. As a matter of principle, there is no distinction between Help information presented in one format (text) as opposed to another (audio or even multimedia). Consumers will take multimedia Help for granted in the near future. Nothing in the NSPR draws a distinction between Help information presented one way rather than another. Faced with that kind of uncertainty, Section 1 would require Microsoft to attempt to design and develop new operating systems so that *no part of the operating system relied*

on any other part that might be considered middleware and thus, might be removable under Section 1. Here again, the resulting operating system—bloated and slowed with the duplicate code needed to eliminate cross-dependencies—would have only a small subset of the functions of Windows today, a subset that could be designed so that none of the pieces necessary to its proper operation can be removed before it reaches customers.

80. Going forward, nearly any new feature that Microsoft might contemplate providing in a new version of Windows would likely be regarded as "middleware" under the extremely broad definition of "Microsoft Middleware Product" provided in Section 22.x.ii. That is true because for any platform feature that Microsoft might add to Windows, there is likely to be at least one other firm in the industry that provides some similar software that exposes at least a few APIs, thereby turning the Windows feature into a removable "Microsoft Middleware Product" under Section 22.x.ii. Section 1 thus could prevent Microsoft from adding significant new features to future versions of Windows in a way that would ensure that those features reach consumers and can be relied on by software developers.

81. By reducing Windows to some undefined "core operating system," the NSPR would turn back the clock on Windows development by about ten years and effectively freeze it there. While operating system and platform competitors continue to add new cutting-edge features to their products, Windows would be stuck until 2012 (the term of the NSPR) in circa 1992-era functionality.

3. Testing to Assure Product Quality

82. A third key aspect of Microsoft's successful development and promotion of its Windows operating system products that is imperiled by the NSPR is the

work Microsoft does to test thoroughly each operating system product it releases. Our testing work is not glamorous, but it is essential to the success of Windows in the marketplace.

83. Modern operating systems such as Windows are enormously complex products. Windows XP, for example, contains literally tens of millions of lines of software code, developed over the course of many years by thousands of engineers at Microsoft. In the course of creating any complex software product, bugs and other problems always arise. The only way to find these problems is to test thoroughly the product, trying out all its capabilities in a rigorous, organized way under various scenarios.

84. The process of testing Windows is greatly complicated by the fact that Windows must work with hardware and software from thousands of other companies. Focusing just on PC components, Windows must work well with various microprocessors, motherboards, video cards, audio cards, graphics cards, hard drives, floppy drives, CD and DVD drives, modems, and so forth. Windows must also work well with a wide range of peripherals, including printers, monitors, keyboards, pointing devices, joysticks, cameras, scanners, camcorders, digital music players, handheld devices, memory card readers, security devices and more.

85. We must test to ensure that Windows correctly performs the functions that those products expect from it. We must also test to ensure that the interaction of those products with Windows and with each other does not cause problems. For example, an application may make changes to a PC that inadvertently causes another application to malfunction.

86. To release a quality product, we test Windows against various combinations of the wide range of hardware and software products. Our testing effort is massive. In fact, to create a new version of Windows, we generally put more testers than developers on the project. Put differently, at least half of the work we do to produce a new version of Windows involves testing the product, not writing the software. In the case of Windows XP, about 1,900 testers were involved in testing Windows XP for about 20 months. In round numbers, we subjected Windows XP to more than 5,000,000 hours of testing at a cost of roughly $500 million.

87. We also subject each new release of Windows to an exhaustive external testing process, subjecting it to use in the field and collecting feedback from customers both on features and performance. Microsoft distributed nearly 500,000 copies of beta versions of Windows XP for testing. This "beta testing" process also reveals bugs, typically tens of thousands of them. The final months of the development process for a new release of Windows are devoted almost entirely to identifying bugs through internal and external testing and fixing them.

88. The testing burden does not end even after a new version of Windows is released. Upon release, customers inevitably find still more bugs in the product, some that may be quite obscure yet important to particular customers. Microsoft undertakes to fix many of these for specific customers by issuing code updates in the form of QFEs, which stands for "Quick Fix Engineering." In the case of Windows 2000, for example, we provided customers with approximately 1,600 QFEs within the first year of its release.

89. In Windows XP we introduced an innovation that enables Microsoft and ISVs to draw upon real-world data in order to discover and fix problems in the interoperation of Windows, applications, third party drivers and other software installed on a PC and hardware. When a problem occurs, the operating system will in most cases display a message inviting the user to send an anonymous "error report" to Microsoft containing details on the nature of the problem. (The report is transmitted over the Internet using the HTTP support we built into the operating system.) We get a massive volume of error reports, enabling Microsoft to see which problems come up most frequently and dedicate resources to fixing them. We are working on improvements to Windows that will enable us to fix problems that users report without any further action by them, completing an excellent feedback loop between Microsoft and its customers. In addition, we recently created a portal where ISVs and IHVs may access the error data that pertains to their products, enabling them to identify and fix problems as well.

90. As I explain more fully in Section III, it would be utterly impractical for Microsoft to undertake the level of testing and problem solving described above if the NSPR were in place. As it stands today, Microsoft typically conducts testing against two primary versions of a new Windows desktop operating system, one of which (the "professional" version) is a superset of the other (the "home" version). Microsoft could not assure the level of quality that it provides today if we were required to test Windows on the assumption that OEMs or others might run Windows in any of 1,000 or more variations, with entire blocks of interdependent code removed altogether. Nor could we effectively diagnose and fix problems that arise when PCs are actually in use.

B. MICROSOFT'S PROMOTION OF INTEROPERABILITY

91. Over the years, Microsoft has worked to ensure that products from a variety of companies work well together. Indeed, I believe that Microsoft has done more to promote interoperability among computer products than any other company in history.

92. Literally tens of thousands of hardware and software products interoperate very well with Windows today. Such interoperability extends not only to products in the PC ecosystem, but also to products in the larger computing ecosystem, such as servers that run UNIX operating systems and other computing devices that do not run Windows. The vast majority of large organizations in the world run heterogeneous computing systems, consisting of various combinations of (i) mainframes, (ii) servers running a variety of Microsoft and non-Microsoft operating systems and hosting powerful database software from Oracle and others, and (iii) PCs running Windows. These systems work well today, and they contribute powerfully to economic productivity.

93. Interoperability across disparate computing products does not happen by accident. Interoperability is a two-way street, requiring a lot of hard work between companies that want to build interoperable products. As discussed below, Microsoft devotes enormous efforts to promoting interoperability between a wide variety of products and Windows. These efforts include our development and broad licensing of the Windows platform (described above) and our disclosure of vast amounts of technical information about Windows—information that we provide to our direct competitors, such as Sun.

94. We work to enable interoperability because the market demands it. Proof of our success is provided by the large number of products that interoperate with Windows today, including server software from Sun and Novell and, of course, tens of

thousands of Web sites that run on various versions of UNIX and are accessible from Windows-based PCs.

95. Given the complexity of modern computing technology, the speed with which it changes, and the large number of products that interoperate with Windows, issues will always arise concerning the precise manner in which one product interoperates with another. Typically, interoperability among disparate products may be achieved multiple ways, and we attempt to work through these issues and provide the marketplace with the interoperability it values one way or another.

96. As in any area of technology, room for improvement always exists, and Microsoft is working hard to continue to enhance interoperability in a wide variety of ways. Among other things, we are continuing to build support for industry-standard protocols and other industry-standard technologies into Windows, thereby enhancing interoperability between Windows and non-Windows operating systems. For this and other reasons, Windows XP provides greater opportunities for interoperability than did Windows 2000, and Windows 2000 provided greater opportunities for interoperability than its predecessor, Windows NT 4.0. The NSPR would frustrate our efforts in this regard by authorizing OEMs and others to remove technologies, including support for basic Internet standards like HTML and HTTP, from Windows that are essential to broad interoperability.

97. Microsoft's .NET initiative is directed at enabling broad interoperability across wired and wireless networks, regardless of operating system, device or programming language. I believe that no other company in the industry shares as broad a vision or is devoting as many resources as Microsoft is to making such interoperability a reality.

98. In the discussion below, I briefly identify four important ways in which Microsoft promotes interoperability today. The NSPR threatens all of them. As I will explain in Section III, many provisions of the NSPR would hinder, rather than promote, interoperability.

1. Development of APIs and Protocols

99. One of the primary ways in which Microsoft promotes interoperability is through the development of technology in Windows that supports various APIs and protocols. Using APIs exposed by Windows, ISVs can create software programs that work very well with the operating system. Literally millions of individual software developers create programs that run on Windows. The vast majority do so without any direct interaction with Microsoft, drawing on the enormous volume of technical information made available by Microsoft and on high-quality development tools offered by Microsoft and others, such as Rational Software and Borland.

100. In addition, the rich set of APIs provided by Windows can be used to create software programs that act as bridges between Windows and non-Windows operating systems. For example, Novell's technology model for its flagship NetWare operating system products entails installation of client software on Windows and non-Windows PCs to facilitate interoperation between PCs and servers running NetWare.

101. A protocol is a method of communicating information across a network. Morse Code, a series of dots and dashes that represent letters of the alphabet, is an example of a very simple protocol outside the context of computing.

102. Using protocols supported in Windows, or other protocols that developers can add to Windows, information can be exchanged between Windows-based

PCs and computers running other operating systems. For example, as I briefly alluded to above, the Web browsing software in Windows includes a system service known as WININET that enables any developer to create programs that interoperate with Windows using the industry-standard HTTP protocol. To take one example, personal finance software such as Intuit's Quicken and Microsoft Money use the HTTP support built into Windows to transmit financial information, such as account balances and stock prices, to and from computers at banks and brokerages (computers that often run a version of UNIX).

103. As shown in Appendix A, Microsoft has steadily increased the number of protocols supported in Windows that it makes available to developers, enhancing interoperability with each new release. We will continue to do so in the future.

104. I understand that the non-settling States believe the Court should enter a disclosure remedy in this matter directed at "permit[ting] rival software to achieve interoperability with Microsoft software . . . ." (Plaintiff Litigating States' First Amended Proposed Remedy, March 4, 2002 at 12.) Section 4 of the NSRP would mandate an extraordinarily broad disclosure of technical information concerning Windows interfaces and protocols. Yet, all the disclosure imaginable will do little to promote interoperability if, as Sections 1, 2 and 7 provide, OEMs and others are free to remove the software that supports the disclosed interfaces and protocols from Windows. If software code is removed, the APIs and protocols supported by that code are removed as well.

2. Disclosure of Technical Information

105. I believe that Microsoft discloses more technical information about its products than any other software company. In fact, Microsoft actively "evangelizes" Windows by providing developers with technical information about it and encouraging

them both to use Windows and to provide feedback to Microsoft. Microsoft's documentation and evangelization of technical information concerning Windows are key reasons why Windows is successful.

106. Detailed information concerning the Win32 API set and other technical aspects of Windows are widely available in bookstores. Developers writing programs to work with Windows can choose from among development tools offered by Microsoft or others. These tools usually include relevant portions of Microsoft's Windows Platform Software Development Kit (or "SDK"). The Windows Platform SDK is a set of tools and documentation that Microsoft broadly licenses to software developers (at little or no cost) to assist in creating software that runs on Windows.

107. An important method by which Microsoft disseminates technical information directly to the developer community is through the Microsoft Developer Network ("MSDN"). (The MSDN Web site is at http://msdn.microsoft.com/default.asp.) MSDN contains an enormous body of technical information about Windows and other Microsoft products. A great deal of information is available free of charge on MSDN, and additional information useful to professional developers is available with the purchase of low-cost subscriptions to the service. Access to MSDN is available to any developer, including Microsoft's direct competitors. For example, Sun Microsystems has more than 100 subscriptions to MSDN.

108. Through the free MSDN Web site, any software developer can receive extensive documentation about Windows APIs. MSDN also includes other forms of technical information and resources for developers, such as SDKs for Windows and other products, Device Driver Kits (which assist in writing software that interfaces with

hardware devices), royalty-free sample applications (demonstrating techniques for software development using Microsoft technologies), and access to the Microsoft Developer Knowledge Base (a database of tens of thousands of articles containing bug reports and "workarounds" relating to Microsoft development products). MSDN is also where developers go to obtain full copies of shipping products and beta code for products that are still under development.

109. MSDN materials also are distributed to developers on CD-ROMs and DVDs. Subscribers to MSDN receive hundreds of disks worth of software and information, each year, all fully indexed and searchable.

110. MSDN is very popular with developers. More than 4 million developers use the Web site each month. Last year, developers generated nearly a half billion page views (one developer viewing one page) on MSDN.

111. Another important way in which Microsoft distributes technical information about its products is through various conferences, which are open to all and designed to meet the needs of various constituencies in the software developer community. About once every twelve to eighteen months, for example, Microsoft holds a large Professional Developers Conference (PDC). Microsoft's PDCs are multi-day events featuring a wide variety of seminars (about 200 this year) designed to educate developers about the latest technologies under development at Microsoft and provide them with insight into Microsoft's future platform plans.

112. Our most recent PDC was held in October 2001, and nearly 7,000 developers attended. I give the keynote speech nearly every year, focusing on important technology trends that we believe developers should start taking advantage of in their

products. This year we focused on Web services and Microsoft's .NET platform for building such services.

113. Other important developer events hosted by Microsoft include Developer Days and Tech·Ed. Typically held annually, Developer Days consists of regional presentations concerning technical aspects of Windows and other Microsoft technologies. Microsoft's most recent Developer Days events were held in 32 cities in the United States in early November 2001. Dozens more were held overseas.

114. Tech·Ed is another multi-day conference, aimed at a broader base of software developers, that provides extensive technical information about Microsoft products. Our most recent Tech·Ed was held April 9-13 in New Orleans, and featured more than 200 sessions on how to build products that take advantage of a wide range of Microsoft products, including Windows.

115. Microsoft's PDCs, Developer Days and Tech·Ed events are open to all developers, including Microsoft's competitors.

116. For the reasons set forth in Section III, the utility of Microsoft's disclosures of technical information described above would be greatly reduced if the NSPR were in effect. Among other things, Microsoft would be obligated to devote massive resources to documenting thousands of internal interfaces within Windows that are neither intended nor tested for use by external developers rather than focusing upon delivering documentation that developers actually need to make products that work well with Windows. Even more fundamentally, the NSPR would greatly reduce Microsoft's incentive to invest in innovation, so that there would be fewer innovative technologies in the future that would be of any interest to developers.

3. Microsoft's Open Review Process

117. Another important way in which Microsoft discloses technical information about its products—and receives valuable feedback—is Microsoft's Open Review Process. Through this process, Microsoft provides alpha and beta code to developers interested in particular technologies and obtains extensive feedback concerning how the technology should be developed to best meet their needs.

118. The Open Review Process begins even before Microsoft starts developing many new technologies. The first step is Microsoft's presentation of a technical proposal for a new initiative to a group of a few dozen developers who are likely to be interested in using the technology. After collecting their feedback, Microsoft conducts a "design preview," which is a more in-depth look at the technology plans, to a somewhat larger group of developers. The next step is a "design review," which provides still more detail about the project under development and will usually include actual code built to the specifications under discussion. Developers are invited to use the code and provide additional feedback both as to design and as to Microsoft's implementation of the design. Finally, Microsoft prepares a final specification, which typically will be posted to MSDN and made available to the developer community at large via SDKs and the like.

119. Our development of innovative new directory and management services in Windows 2000 provides a useful case study of the Open Review Process. In August 1995 and February 1996, four years before Windows 2000 was released, we conducted Design Previews of early specifications for our planned directory and management services in the new operating system for about 80 to 120 developers. The next month, we provided updated specifications to about 300 developers who attended the relevant seminar at our Internet Professional Developers Conference. In July 1996, we

provided early beta code and updated specifications to about 90 developers, and in November, we provided about 3,000 developers with a preview (pre-beta) version of Windows NT 5.0 (as Windows 2000 was then known) that included our directory and management services. We held another Design Preview in May 1997 (for about 180 developers), released Beta 1 of Windows NT 5.0 to 6,000 developers at our Windows NT 5.0 PDC in September 1997, and conducted yet another Design Review for 65 developers in November 1997. By the time it was released commercially, Microsoft had distributed hundreds of thousands of beta test copies of Windows NT 5.0 worldwide.

120. By the time Windows 2000 was released in February 2000, the process of reviewing, testing and providing feedback on its directory and management services that began with a few dozen developers 4½ years earlier had grown to thousands of developers outside of Microsoft.

121. Microsoft would be prohibited from providing information and obtaining feedback via the Open Review Process if the NSPR were in effect. Section 4, read in conjunction with the definition of Timely Manner (Section 22.pp), appears to require that Microsoft disclose technical information to the industry generally (ISVs, IHVs, etc.) at the same time that information is "disclosed to any third party." It is not practical to disclose information concerning a new technology to the industry at large before specifications for the new technology have even been prepared, much less before significant development work has been undertaken.

4. Testing Interoperability

122. Testing is essential to ensuring that products interoperate with each other. No other software company engages in product testing that approaches the scope and

complexity of Windows testing because no other product seeks to provide compatibility with as many third-party hardware and software products as Windows. A very substantial portion of all the testing we do on Windows is directed toward testing the interoperation of Windows with other software and hardware products, including non-Microsoft server operating systems, such as UNIX. Even in a theoretical world of perfect information flow between developers working on two products, testing would be essential to find bugs or other problems in their implementation (the code they actually wrote) as the two products interact with one another.

123. As I described above, it would not be feasible to test all the many variations of Windows that OEMs and others might elect to ship under the NSPR. Absent such testing, bugs will not be found, quality will inevitably decline and interoperability will suffer.

C. Intellectual Property Rights

124. Microsoft is an intellectual property (IP) company. We have no factories of any consequence or natural resources. Indeed, we have no physical assets of any kind that are important to the success of the company. Our products instead consist almost entirely of information we create—primarily instructions to microprocessors on how to perform various functions. Those instructions are reflected in source code (written by Microsoft software developers) and object code (a machine-readable form of the code written by developers, created by running that source code through a "compiler" program). The source code, object code and related information are protected to varying extents by copyright, patent and trade secret law.

125. Some of Microsoft's most valuable intellectual property is the trade secrets that may be learned by reviewing the source code to our products. Unlike patent and copyright rights, which survive any disclosure, trade secrets are forever lost once revealed.

126. Absent intellectual property protection, there would be little reason to invest in developing software. Software can be copied and distributed quickly and at near-zero marginal cost. The very low marginal cost of copying software means that if two firms have rights to the same piece of software, the price of that software quickly tends to be competed down to zero.

127. Even absent literal copying of software code, software innovations developed by one firm can be implemented by competitors writing their own code. The more information one firm has about a competitor's product, the easier it is to copy the key features and other innovations of the product. In the software industry, some information about competitors' products is available, and other information is protected by IP laws. If Microsoft's competitors were permitted to implement many of Microsoft's innovations in their own products without regard to Microsoft IP rights, Microsoft would have little it could uniquely offer the marketplace. No firm can do unique R&D in the software industry absent significant IP protection for its work.

128. As I will detail on a provision by provision basis in Section III, the NSPR would strip Microsoft of a broad range of very valuable IP rights— IP rights that, together with our employees, constitute Microsoft's most important assets and thus underpin the company's market capitalization of approximately $300 billion. The NSPR directly targets the two most important products at Microsoft, Windows and Office. These

two products collectively account for approximately two-thirds of our revenue. Very briefly, the NSPR would:

- Transfer important IP rights in Windows from Microsoft to anyone who wishes to license 10,000 copies, allowing them to remove important parts of Windows before delivering it to the marketplace and to modify it in many other important ways while still marketing the resulting operating system to consumers under the "Windows" trademark;
- Impose price controls on Windows that would effectively penalize Microsoft for investing in improvements to Windows *and* reward third parties who impaired the product by removing functionality;
- Grant equal rights to everyone in the industry to some of the most innovative work Microsoft has ever done, *i.e.*, the development of Microsoft's industry-leading Web browsing software, in which we have invested more than $750 million;
- Grant three companies full licenses to Microsoft's Office technology—as it exists today and all improvements we may make for ten years—and the right to use that technology to offer Microsoft's Office technology on various platforms, including clones of Windows, on the basis of an "auction" that would greatly devalue Office by vesting four companies with rights to the same technology; and
- Require Microsoft to provide the industry at large with rights to a vast range of IP concerning the inner workings of Windows and Office, including access to actual source code, that would be useful to competitors seeking to emulate Microsoft's innovations in Windows.

With the exception of the Office "auction," the NSPR would provide all of this IP to Microsoft's competitors entirely free of charge. In fact, Section 4 is so sweeping (especially when read in conjunction with the defined terms) that nearly anyone in the industry could demand that Microsoft provide it with royalty-free access to nearly any

important Microsoft platform technology under a claim that access to that technology was necessary to enable interoperability.

129. Microsoft is investing $4.5 billion in R&D this year. If the NSPR were in effect, it would not be sensible for Microsoft to invest nearly so heavily in Windows and Office innovation. There would be little reason to invest in R&D under the NSPR because Microsoft would become simply a technology provider to its competitors, receiving little economic return for our past work or for new work on Windows or Office for the next ten years.

III. THE NSPR

130. I believe that the NSPR would greatly reduce Microsoft's incentive and ability to develop and deliver new technologies to the marketplace. The consequences would be three-fold. First, all those who build upon or otherwise benefit from Microsoft's heavy investment in developing new technologies—OEMs, ISVs and the businesses and consumers who use our software—would be harmed. With the loss of the positive feedback benefits provided by Windows, the marketplace would experience higher prices and less innovation.

131. Second, Microsoft would be greatly devalued as a company. Microsoft's market capitalization is based on the market's well-founded belief that Microsoft is on a path to deliver a wide range of breakthrough technologies that will generate new sources of revenue.

132. Third, competition would be reduced not only in operating systems, but also in key product categories where Microsoft is the strongest challenger to incumbent leaders, such as online services (where AOL Time Warner leads), handheld devices (where

Palm leads) and game consoles (where Sony leads). Indeed, in the important area of server computing—both hardware and software—the strongest competitive challenge to incumbent, high-price UNIX vendors such as Sun is the PC model of multiple, competing OEMs building upon a standard, widely licensed and attractively priced operating system, such as Windows 2000 Server.

133. This section of my testimony describes the NSPR on a provision by provision basis, explaining why the remedies proposed by the non-settling States would reduce Microsoft's development and delivery of new technologies. I begin in Section III.A by identifying three overarching problems that run throughout the NSPR. Section III.B then discusses each substantive provision of the proposed remedy.

A. Overarching Problems

134. There are three key aspects of the NSPR—the breadth of the covered product categories, the vagueness and ambiguity of many of its most important provisions, and the feasibility of complying with various of its requirements—that are especially alarming to me. I believe that these aspects of the NSPR would make it extremely difficult for Microsoft to understand the requirements of the NSPR, to comply fully with the requirements it does understand, and to continue to deliver new technologies to the marketplace. In short, the practical effect of the NSPR would be to cripple Microsoft as a technology company.

1. Breadth of Regulated Product Categories

135. I understand that this lawsuit concerns competition in operating systems for PCs. The NSPR, however, imposes sweeping restrictions on Microsoft without regard to the product category at issue in this lawsuit. I provide several examples below.

a. "Any Microsoft Product"

136. Section 6.c provides that Microsoft may not enter into any agreement in which any third party agrees to distribute, promote or use *any Microsoft product, service, feature or technology* exclusively or in a fixed percentage. Yet such agreements are common throughout the economy and are often vitally important to creating economic value. For example, game console vendors compete in part by attracting ISVs to offer popular games exclusively for their game console. Many games are available only on the leading Sony PlayStation, not on Microsoft's new Xbox or Nintendo game systems, and Sony promotes the PlayStation on that basis. Microsoft needs the ability to promote Xbox on that basis as well if we are to compete in the game console business.

137. Similarly, simple advertising deals often entail a third party's commitment to "promote" the advertised product exclusively or in a fixed percentage. For example, MSN might enter into an arrangement to be the "exclusive sponsor" of an event presented on the Internet, such as the release of a new album from a music artist, or another Web site might agree to run MSN advertising on some percentage of its Web pages for a specified period of time. In either case the third party would be promoting MSN exclusively or in a fixed percentage. Such routine business transactions would be prohibited by Section 6 of the NSPR.

138. Similarly, Section 8 prohibits Microsoft from taking any action that directly or indirectly adversely affects a third party based on its use, distribution, promotion, support or development of *any non-Microsoft product, service, feature or technology.* Section 8 is very broadly worded: it appears to prohibit Microsoft from competing in any product category because nearly any act of competition could be said to *directly or indirectly adversely affect* a competitor.

b. Middleware Definitions

139. The "Middleware" definitions in Sections 22.w and 22.x are at once extremely broad and, as discussed below, vague and ambiguous. Since the meaning and effect of many provisions of the NSPR turn on the various definitions of "middleware," the scope of these terms is critical.

140. I understand that the trial primarily concerned Microsoft's efforts to compete with Web browsing software from Netscape and, to a lesser extent, with various aspects of Sun's Java. Plaintiffs alleged that Netscape Navigator and Java had the potential to develop into software development platforms that would run on multiple PC operating systems. If a sufficient number of ISVs wrote applications that drew on capabilities provided by these platforms, Plaintiffs argued, consumers would have less interest in running Windows and might use non-Microsoft operating systems under their Web browsing (or Java) software layer. As Plaintiffs noted, I expressed concern in my 1995 memorandum on the rise of the Internet that Netscape was "mov[ing] the key API[s] into the client to commoditize the underlying operating system." (In evidence as GX 20 and offered here as DX 1487.)

141. My understanding is that the Court of Appeals' liability determination turned largely on Netscape's ability to distribute Navigator (and with it, Java, according to plaintiffs) in two channels of distribution: the OEM channel and the IAP channel.

142. The potential competitive significance of Netscape Navigator and Sun's Java turned on two key attributes of those programs, neither of which is shared by the many software categories deemed to be middleware under the NSPR.

143. *First*, Navigator and Java supposedly had the potential to become general-purpose software development platforms. In other words, under plaintiffs' theory, Navigator and Java had the potential to provide a full set of system services, via APIs, so that developers could write applications such as word processing software, spreadsheets and personal finance software to them as platforms. If the platforms were good (a big "if" because developing good platform software is hard), ISVs might well write applications to run on Navigator or Java rather than Windows.

144. At the time, we were struck by the statement of Marc Andreesen, a co-founder of Netscape, that Navigator would reduce Windows to little more than a "poorly debugged set of device drivers." That statement meant that Navigator would strive to compete directly with Windows, providing the key APIs to developers and leaving Windows simply to handle PC interaction with peripheral devices such as monitors and printers. With the benefit of hindsight, we now know that Netscape never did the hard work needed to become a PC applications platform, but it could have and AOL certainly has the resources to pursue this strategy today. Similarly, Sun provided a quite broad set of system services (through its Java class libraries) to enable the development of a broad range of applications on the Java platform. For example, Corel produced a beta version of its office productivity suite, complete with word processor, database, presentations software and so forth, to run on the Java platform. Corel's Java-based office productivity suite did not succeed due to various shortcomings in the client-side Java platform.

145. *Second*, Navigator and Java both ran on Windows and other PC operating systems and thereby competed with similar system services *in* Windows to the extent they provided APIs to developers building PC applications.

146. The NSPR does not reflect either of the attributes of Navigator and Java that were so important to the competitive challenge they potentially posed to Windows. The elements that define "Middleware" in Section 22.w are listed below. None of them significantly bounds the types of software that would fit the NSPR's definition of middleware. To be "Middleware," software must be:

a) *"provided in the form of files installed on a computer or in the form of Web-Based Software"* All software consists of files installed on a computer (whether that is a client or server computer), so this is no limitation at all.

b) *"operates directly or through other software . . . by offering services via APIs or Communications Interfaces."* Nearly any software that runs on a computer can offer services via APIs or Communications Interfaces.

c) *"and could, if ported . . . . enable software products written for that Middleware to be run on multiple Operating System Products."* Many software programs "could" be ported to multiple operating systems and thereby enable any services it provides to be available cross-platform. It is just a matter of doing the necessary work.

(Section 22.w.)

147. In short, the definition of "Middleware" in the NSPR appears to be very close to "any software that exposes a few APIs." (In fact, the non-settling States technical expert, Professor Appel, and their Rule 30(b)(6) witness, Mr. Greene, both testified that the presence of a *single* API can be enough to make a piece of software "Middleware." (Deposition of Andrew Appel, February 2, 2002 at 77; Deposition of Thomas Greene, March 12, 2002 at 61.) Given that nearly any software can expose a few APIs, that definition is just one step short of "all software." But not all software that exposes a few APIs presents any real competitive challenge to a desktop operating system such as Windows. Rather, only software that runs on desktop versions of Windows and

other operating systems and competes with Windows by serving as a general purpose development platform (or that reasonably could become such a platform) presents a real competitive challenge.

148. Section 22.w of the NSPR includes as "Middleware" many categories of software that do not serve as general purpose development platforms and thus do not present any significant competitive challenge to Windows. For example, "Handheld Computing Device synchronization software" is just that—a relatively simple program that enables a handheld device such as a Palm Pilot or a Pocket PC to synchronize information such as schedules and email with scheduling and email software on a PC. That is a highly specialized function, not a development platform for building software applications. The same holds true for "calendaring systems." If calendaring software exposes any APIs, those APIs are likely to be useful for a single, specialized purpose: to provide calendaring functionality in another software program.

149. Other than Internet browsers, network operating systems and the Java virtual machine, none of the categories of software listed as "examples" of Middleware in Section 22.w reasonably could become a general purpose development platform. None provides a set of key APIs that could commoditize Windows. Instead, each is directed at specialized functionality that provides at most a very small subset of the functionality of a general purpose platform.

150. Section 22.w is also very broad because it is not limited to software platforms that run on a PC and thus can provide a substitute for PC operating system functionality. For example, Section 22.w states that a "network operating system" is an example of middleware, but such products run, by definition, on servers, not on PCs.

(Novell is the only company I know of that markets what it calls a "network operating system.") For many years to come, however, the thousands of applications that run directly on Windows-based PCs today will continue to run on PCs. For that reason, server operating systems, set top box software, and other software that doesn't run on PCs will not commoditize Microsoft's PC operating system software.

151. The definition of Microsoft software that would constitute "Middleware" under the NSPR is similarly unbound. Under Section 22.x, a "Microsoft Middleware Product" is simply any software offered by Microsoft that provides functionality similar to "Middleware." Given the breadth of Section 22.w's definition of "Middleware," that could mean nearly any Microsoft software. Indeed, it is a safe bet that for nearly any Microsoft software that exposes APIs (including any feature of Windows), there will be programs from other software vendors that could be said to provide similar functionality.

152. Section 22.x is not limited to software included in desktop versions of Window, even though the case primarily concerned Microsoft's decision to integrate Web browsing software into its PC operating systems. To the contrary, Section 22.x.ii(1) defines as a "Microsoft Middleware Product" any "Middleware" that Microsoft has distributed *separately* from an Operating System Product in the past three years (without regard to whether that software is also distributed as part of Windows). *Nearly all Microsoft software is distributed separately from an Operating System Product*. (Section 22.x.ii(2) then picks up all software *within* Windows as well.)

153. Section 22.x.i calls out Microsoft Office as an example of middleware. Microsoft Office is the most important application that Microsoft offers. It is

not part of Windows. Furthermore, neither Microsoft Office nor the programs with which it competes, such as Corel WordPerfect or Sun StarOffice, are general-purpose development platforms. Microsoft has put a lot of effort into exposing Office functionality to developers via APIs, but that functionality relates solely to business productivity, such as customized word processing or specialized financial reports, not to operating system platform functionality. Office does not provide operating system functionality; it calls into Windows for such functionality.

154. With respect to software *in* Windows, Section 22.x seems to sweep within its ambit nearly any part of Windows that exposes functionality through one or more APIs—which is to say, most of the product.

155. Another key term exacerbating the breadth of the NSPR is the definition of "Microsoft Platform Software" set out in Section 22.y. The term is *not* a proxy for "desktop versions of Windows." The term also encompasses any "Microsoft Middleware Product," which includes *server* versions of Windows, Microsoft Office, the Exchange email server, *and any other software offered by Microsoft, running on any kind of computing device, that provides functionality similar to "Middleware" offered by a competitor.* As shown above, that could be almost anything under the broad definition of "Middleware."

2. Vagueness and Ambiguity

156. The problems that would result from the extremely broad scope of the NSPR are compounded by the fact that many key provisions are vague and ambiguous. That is of particular concern to me because I want to be certain that Microsoft understands

what its obligations are when this lawsuit is concluded so that we can comply fully with the final judgment.

157. Clarity is especially important because so much of the NSPR pertains to engineering decisions. If the final judgment is not clear, we will not know how to build new products that comply with the judgment.

158. In addition, any decisions we make about the meaning of ambiguous provisions will undoubtedly be scrutinized by Microsoft's competitors, leading to one controversy after another, some of which may come before this Court. Absent clarity, the Court will lack any clear basis for adjudicating such controversies. From Microsoft's perspective, on-going controversy would be a very bad outcome. Our objective in trying to settle the case, and our objective now, is to arrive at a set of understandable rules that will govern our operating system business so that we can better avoid protracted and contentious legal confrontations and focus on building great software.

a. Internal Inconsistencies

159. Several provisions of the NSPR appear to be inconsistent with one another, either in their command to Microsoft or in their intended purpose. I am concerned that this will lead to difficult questions of interpretation, particularly when the NSPR is read in view of its direction that "[a]ll of the provisions of the Final Judgment . . . will be interpreted broadly consistent with its remedies purpose . . . ." (Section 21.c.)

160. For example, Section 4 imposes broad obligations on Microsoft to disclose a wide range of technical information concerning interfaces in Windows for the stated purpose of promoting interoperability with Windows. Yet Sections 1 and 2 authorize third parties to remove large portions of Windows, including software that supports the

interfaces that must be disclosed under Section 4. If OEMs remove software that supports APIs, disclosure concerning those APIs is not going to promote interoperability. The APIs will not work if the software is removed and developers will be much less likely to use them if that is a possibility.

161. Similarly, Section 16 establishes circumstances under which Microsoft is obligated to comply "fully" with certain industry standards in Windows (and other products). Once again, Microsoft would be unable to provide any assurance that its operating systems actually comply with industry standards (so that developers writing applications for Windows could rely upon those standards) if third parties were free to remove the software that implements the standards. If an OEM exercises its right under Sections 1 and 2 to remove Microsoft's Web browsing software, for example, Windows will no longer comply with the HTTP standard (and other Internet-related standards), in apparent violation of Section 16.

b. Middleware Definitions

162. Many of the key terms in the NSPR are defined far more broadly than how the terms are ordinarily used in the computer industry, potentially leading to ambiguity when attempting to apply the terms to our business. Even the term "Microsoft" has been given a special definition that seems to sweep within it nearly any company. ("Microsoft" means Microsoft's "assigns," including "any transferee or assignee of any . . . ability to license the Intellectual Property referred to in this Final Judgment." Many sections of the NSRP grant third parties the right both to (i) *take* licenses to Microsoft's Intellectual Property and (ii) further *grant* licenses to that Intellectual Property to others.)

163. An important example of vagueness and ambiguity is the NSPR's definitions relating to middleware. There is nothing in Section 22.w that provides any technical or legal distinction between the examples of things that are "Middleware" (such as synchronization software) and those that are not, namely, disk compression and memory management software.

164. Even more troubling is the definition of "Microsoft Middleware Product" in Section 22.x. The NSPR would impose a number of very significant design and disclosure obligations upon Microsoft relating to software—labeled as "Microsoft Middleware Products"—in Windows. However, *the NSPR provides no rule for determining which code within Windows constitutes the various "Microsoft Middleware Products" identified by NSPR* (much less a rule for determining what code will constitute other "Microsoft Middleware Products" under Section 22.x.ii). That is a serious omission because Microsoft's obligations under Sections 1, 2 and 4 depend entirely on knowing which software code is and is not included within specified middleware categories. If we do not know what code constitutes the various "Microsoft Middleware Products" we will not know how to build operating systems that comply with Sections 1 and 2 or how to make the required disclosures under Section 4.

165. The problem is compounded by the fact that the NSPR appears to define "Middleware" at a very granular level: if even a single API can render a program "Middleware," then small programs or even individual files or parts of files within Windows might well be regarded as "Microsoft Middleware Products" under the NSPR.

166. This is a major problem because the degree of "granularity" (size of modules) that will be applied in defining what software constitutes "Microsoft Middleware

Products" is determined by design decisions made by *third party software developers* (because a "Microsoft Middleware Product" is defined to be software similar to whatever "Middleware" a Microsoft competitor may create). Functionality that Microsoft might well regard to be a feature of Internet Explorer, for example, might well be offered as a standalone product by a third party, rendering portions of Internet Explorer "Microsoft Middleware Products" in their own right.

167. Of course, when designing new operating systems, Microsoft will not have complete knowledge of all the software programs available in the world that may be deemed "Middleware," much less software programs that may be released after Microsoft makes key design decisions, committing itself to a particular development path based on decisions concerning where the "Microsoft Middleware Products" lines would be drawn.

168. The central problem is that the NSPR—and much of the testimony of the non-settling States' witnesses—proceeds on a faulty premise. The NSPR and the non-settling States' witnesses speak of Windows as if it consisted of a small, self-contained and fully functional operating system (which they sometimes refer to as the "kernel" or the "core operating system") and a collection of readily identifiable, add-on middleware products that might just as easily be distributed separately from the operating system. In fact, each version of Windows is designed as a single, integrated product that provides a broad range of functionality. Various parts of the operating system provide functionality to other parts of the operating system (through interfaces that may or not be documented for external use) as well as to developers (through documented APIs). Given the interdependencies among parts of Windows, and the complexity of the product, *there is no*

clear dividing line between where a particular block of "middleware" ends and the rest of the operating system begins.

169. Such integration—an important aspect of product innovation and competition—is essentially the process of breaking down boundaries between formerly disparate technologies. In the past, consumers purchased and installed a separate "spellchecker" application, and ran it when the time came to "spell check" a document. Today, spell checking is just another feature of a word processor that we take for granted. Rather than launch a separate spelling application, word processing software such as Microsoft Word recognizes spelling errors as they occur and corrects many of them automatically with no user intervention. As a result, the "boundary" between the "word processor" and the "spell checker" has largely disappeared.

170. The significance of the NSPR's failure to define the boundary between "Middleware" in Windows and the rest of the operating system is well illustrated by the record of this case. Microsoft repeatedly pressed Plaintiffs to identify which software in Windows they believed constituted Internet Explorer. We never got an answer. Does Internet Explorer include the software that enables Web pages (or any other software) to display information formatted in HTML? Does it include software that supports the HTTP protocol? We think the answer to both questions is plainly "yes," yet at trial Plaintiffs appear to argue that these important files were part of Windows not Internet Explorer. (*See* Plaintiffs' Revised Proposed Findings of Fact at ¶¶ 161.2.2, 154.3.2.)

171. Although the parties devoted considerable resources to litigating the question of what software actually constitutes Internet Explorer, Judge Jackson never resolved the matter (perhaps because Plaintiffs focused upon removing end user access to

Web browsing, rather than Web browsing code itself). The testimony put on by the non-settling States in this proceeding continues to be unclear on this point. For example, Professor Appel testified that when "unbinding" Internet Explorer from Windows, HTTP support should be removed, but FTP support (another protocol for transferring files) should not. Yet both are just protocols for transferring information between two computers and both are supported in every Web browser. What distinguishes them for purposes of determining Microsoft's engineering obligations under Sections 1 and 2(c)? Professor Appel deemed the date they first became popular to be significant, but there is no such test in the NSPR, and in any event Professor Appel did not suggest that a "date" test would apply in determining what other Windows code was part of other "Microsoft Middleware Products." (Trial Transcript, April 10, 2002 at 3091-93.)

172. A clear rule for determining what code in Windows constitutes the various "Microsoft Middleware Products" is essential to application of Sections 1, 2 and 4. To attempt to comply with Sections 1 and 2, we would need to know what software code must be made optionally removable without degrading the rest of the operating system. For example, does removing Internet Explorer entail removing the HTML rendering software in Windows? That is an important question because many parts of the user interface of Windows, including its Help system, rely on that HTML rendering software. If we design an "unbound" operating system that leaves the HTML software in place when an OEM "removes" Internet Explorer, competitors may claim that we have not really enabled the removal of Internet Explorer because HTML software is integral to Web browsing. Yet if we design an "unbound" operating system in which HTML software *is* removed when

Internet Explorer is removed, then others may claim that other parts of the operating system that rely on that HTML software have been degraded (and they would be).

173. In other words, Section 1's instruction to redesign Windows to enable "binary code" to be removed without causing any "degradation" of the product would put Microsoft on the horns of a dilemma as to each middleware removal decision: remove too little code, and be charged not removing the "Microsoft Middleware Product," or remove too much code, and be charged with "degrading" the operating system.

174. The problem of not knowing what software code to make "removable" is compounded by the fact that there are many categories of "Microsoft Middleware Products" in Windows and as to each program falling into one of those categories our engineers would face multiple line drawing questions.

175. Absent a rule delineating the boundaries of "Microsoft Middleware Products," Microsoft also would not know what it would be required to do to comply with Section 4. Section 4.a.ii would obligate Microsoft to disclose the interfaces between "Microsoft Middleware Products" and the rest of the operating system. Obviously our engineers would need to know where the line is between each "Microsoft Middleware Product" and the rest of the operating system in order to disclose the interfaces between them. Furthermore, under the definitions of the NSPR, the disclosure of new interfaces can lead to additional parts of the operating system being deemed a "Microsoft Middleware Product" (*see* Section III.B.4.b), leading to ever larger number of line drawing problems—problems with no answer in the language of the NSPR.

c. Ordinary Business Practices

176. Section 8 provides another important example of ambiguity in the NSPR, made worse by the very broad scope of the provision. Do the non-settling States really mean that Microsoft should not take *any* action that directly or indirectly adversely affects any third party based on the fact that the third party is competing with Microsoft in *any* product category? That would seem to rule out ordinary business practices. For example, if one OEM elects to build handheld devices based on Palm software, and another elects to build handheld devices based upon Microsoft's Pocket PC software, would Microsoft nevertheless be required to provide both OEMs with the same technical information, information about future plans and even trademarks and logos relating to the Pocket PC?

3. Feasibility

177. Many provisions of the NSPR are not feasible. For example, it would not be feasible to design new operating systems to comply with Section 1's requirements that software code be removable in a thousand combinations or more without degrading the operating system. I discuss this point more fully in Section III.B.1 below.

178. Section 2 states that Microsoft must provide "Covered OEMs" and anyone else that licenses 10,000 copies of Windows—even if not an OEM—with "equal access" to a long list of things, including licensing terms, marketing and sales support, product information, information about future plans and so on. Yet, as in any business, our licensing terms must vary depending upon whether the licensee is an OEM or not. For example, the essence of an OEM license agreement is Microsoft's grant of a right to preinstall a copy of Windows on a new PC, and with that comes a series of provisions

relating to supplying end-users with back-up media, product support, etc. If a licensee is not an OEM, many of those provisions will not be relevant.

179. Even among OEMs, Microsoft's license terms will vary depending upon whether the licensee is shipping 10,000,000 copies of Windows annually (as several large OEMs do) or 10,000 copies annually. Similarly, Microsoft might well engage in joint sales calls on corporate customers with an OEM such as Hewlett Packard that ships millions of PCs annually and focuses on corporate sales. It would make little sense to do joint sales calls with a third-party that licenses only 10,000 copies of Windows and may have no entrée into corporate accounts.

180. Section 6.e states that Microsoft may not enter into an agreement in which an IAP or ICP obtains placement in Windows in exchange for an agreement to use any Microsoft technology. Yet agreements to place a third party in Windows almost inevitably entail that party's use of at least part of Windows to make the placement work technically.

B. Section by Section Analysis

181. The following is a discussion of specific problems presented by each substantive provision of the NSPR.

1. Section 1

182. Microsoft would be unable to redesign its operating system products within six months of entry of the NSPR to meet the design specifications set forth in Section 1, even assuming those specifications were rewritten to eliminate their substantial ambiguities. Given the tightly integrated design of Windows and the complexity of the product, Microsoft cannot ensure that "the binary code for each Microsoft Middleware

Product" could be removed without degrading the rest of the operating system. To the contrary, removing the binary code for "Microsoft Middleware Products" *will* degrade the rest of the operating system—every function that depends upon the removed software will fail.

183. I am concerned that Section 1 would therefore require Microsoft to withdraw Windows from the marketplace because it states that Microsoft may not distribute any version of Windows more than six months from entry that does not comply with its terms (subject to the possibility that the Court would later modify its terms). I know that Microsoft could not have developed Windows 95, one of the most successful software programs in history, if Section 1 had been in effect in the early 1990s.

184. Even if it were feasible to build new operating systems that conform to Section 1 design specifications, and Microsoft embarked upon a massive development effort to do so, the resulting products would be far less valuable to users and developers because they would, by definition, not provide a stable, consistent platform for software development. In addition, Section 1's pricing provisions would create strong disincentives for Microsoft to continue to invest in improving its operating system. Faced with the prospect of building less valuable operating systems and reduced reward for doing do, I doubt that Microsoft could motivate talented software engineers to work on operating system development or that it would make sense for Microsoft to continue to invest in doing so. I discuss these points, in turn, below.

a. Feasibility of Code Removal

185. Microsoft designs each version of Windows to be a single product. When designing a new release of Windows, we assume that one part of the product can

draw on the capabilities of other parts of the product. If one part is removed, every other part that relies upon it will malfunction to some extent or perhaps fail altogether.

186. In other words, as with complex software programs generally, there are salutary code interdependencies within Windows. The use of the industry-standard HTML format within Windows provides an example.

187. When Windows presents information in the HTML format it calls on the component of the operating system that handles HTML rendering file called MSHTML.DLL that works in tandem with other files. If there are a dozen places in Windows where HTML information is presented (such as the Help system, the File system, an Internet Explorer window and so forth), Windows will use the same component every time to handle the HTML rendering. Similarly, if we expose the HTML rendering capabilities to developers via APIs (as we do), the APIs will call into the same HTML component.

188. It is highly efficient to use the single HTML rendering engine in the operating system for multiple purposes. From the perspective of developing software, we do not want the Help team, the File team, and various applications teams all duplicating each other's efforts by building their own HTML rendering software. We want a single team to focus on building and testing a quality component that can be used by all groups throughout Microsoft as well as by third-party developers.

189. From an engineering perspective, we reduce the size and improve the performance (speed of operation) of the operating system by using a single component for multiple purposes. It would certainly be a poor design of a commercial software product to put *multiple* copies of a single component in the operating system, as I understand that

Professor Appel has suggested, when one copy of the component is capable of serving the needs of the entire operating system and third-party applications.

190. From a usability perspective, we look for ways to reduce multiple concepts that consumers must learn to use a computer effectively. Thus, for example, when the "Back" and "Forward" and hyperlink model of Web navigation became popular in the mid-1990s, we began to utilize the same concepts in the Windows user interface—using the same software, of course, that provides those capabilities to the Web browsing software in Windows and to third-party developers who build on Windows.

191. If Section 1 were in effect, Microsoft could no longer design operating systems on the assumption that Component A (such as the Help system) could rely on Component B (such as the HTML rendering software) because Microsoft would be obligated to ensure that Component B can be "readily removed" if it were a "Microsoft Middleware Product."

192. Indeed, Microsoft would face an immediate crisis if the NSRP were entered because Section 1 would prohibit Microsoft from distributing any Windows Operating System Product after six months that does not comply with the requirements of that section (unless an extension could be obtained from the Court). We could not comply with Section 1 in six months for the following six reasons.

193. First, as explained above, each of our Windows Operating System Products was designed on the assumption that it would not be deconstructed by OEMs and other intermediaries. There are innumerable interdependencies in the millions of lines of software code that make up each of these operating systems—interdependencies that have accumulated over the course of years of software development. If the "binary code" for any

"Microsoft Middleware Product" is removed, every part of the operating system that relies upon that component will be "degraded." For example, if the binary code for Internet Explorer were removed, then everything in Windows that depends upon the HTML rendering software would fail. There is no getting around this basic fact.

194. Second, as discussed above, Section 1 provides Microsoft with no instruction concerning what software constitutes "the binary code" for each "Microsoft Middleware Product." That is a major problem. Given the integrated design of Windows, it is often not obvious where the binary code for a "Microsoft Middleware Product" ends and the rest of the operating system begins. Without a clear rule for determining what software must be made removable, Microsoft's software engineers will not know on which parts of Windows other parts of the operating system (and third party developers) can rely. Absent that information, our engineers could not take the first step in attempting to build a compliant operating system.

195. Third, Section 1 is ambiguous in demanding that the "unbound" operating system "perform[ ] effectively and without degradation (other than the elimination of the functionalities of any removed Microsoft Middleware Products)." Does this mean that if Internet Explorer were removed, then consumers should be unable to open an Internet Explorer window and browse the Web, but everything else in the operating system (such as the Help and File systems) should work effectively? That would essentially require that all the Internet Explorer binary code be left *in* Windows because that code is used so heavily throughout the operating system.

196. Alternatively, does the language "other than the functionalities of any Microsoft Middleware Product" truly mean that all the "functionalities" provided by

Internet Explorer (in this example) must be removed? That would mean that the many parts of the operating system that rely on those "functionalities" will fail. I am concerned that release of such an "unbound" operating system, full of error messages and other failures, would not be deemed compliant with the NSPR.

197. And what does Section 1 contemplate with respect to the APIs exposed by "the binary code for each Microsoft Middleware Product"? Should the APIs continue to perform after the Microsoft Middleware Product is removed so that applications work properly? Or does Section 1 require that such APIs be removed as well? I understand that the witnesses for the non-settling States disagree among themselves on this question.

198. The non-settling States' Rule 30(b)(6) witness, Mr. Greene, their technical expert, Professor Appel and Sun's Jonathan Schwartz appear to have testified that the intent of Section 1 is to enable OEMs and others to remove the binary code for "Microsoft Middleware Products" and thereby remove the APIs provided by that code. (Deposition Testimony of Thomas Greene, March 12, 2002 at 39-40; Direct Testimony of Andrew Appel at ¶¶ 129-138; Direct Testimony of Jonathan Schwartz at ¶ 158.) Yet, Mr. Greene agreed that removing APIs would be "tough luck" for developers who built software that uses those APIs. (Deposition Testimony of Thomas Greene, March 12, 2002 at 39-40.) Sun's Richard Green disagrees. He interprets Section 1 to require that APIs continue to function even after a "Microsoft Middleware Product" has been "removed." (Deposition Testimony of Richard Green, March 16, 2002 at 323-324.) That would require, of course, that the binary code supporting the APIs, *i.e.*, the Microsoft Middleware Product code, *not* be removed.

199. Put simply, the "perform[ ] effectively and without degradation" language followed by the parenthetical provides no rule for determining what parts of the operating system should break and what should remain functional after the binary code for any particular "Microsoft Middleware Product" is removed.

200. Fourth, even if the above problems were somehow solved, Microsoft would be unable to deconstruct and reconstruct Windows in the manner apparently contemplated by Section 1 given the complexity of the operating system and the breadth of the NSPR's definition of "Microsoft Middleware Product." Section 22.x.i defines ten categories of "Microsoft Middleware Product" in Windows today that must be made "removable" within six months under Section 1. Some of these categories, such as "systems and enterprise management software," would cover multiple components within Windows. Even assuming just ten removable components, however, Microsoft's engineers would need to develop an operating system that could be deconstructed by a third party into any of 2^{10} (or 1,024) possible combinations, while ensuring that each combination "performs effectively and without degradation."

201. Since four Microsoft operating systems, built on two separate code bases (Windows 95, Windows ME, Windows 2000 Professional and Windows XP), are subject to Section 1, we would face the immediate prospect of attempting to re-engineer our products to ensure that any of 4,096 combinations of operating system software would comply with Section 1. This describes our task in the first six months. Although the NSPR is ambiguous on this point, it appears that at some future, unspecified time, Microsoft would be obligated to make "removable" the binary code for all the additional parts of Windows that fit Section 22.x.ii's definition of "Microsoft Middleware Product." If just 20

additional components of Windows must be made removable, Microsoft would be obligated to design our products to ensure that any of more than 4,000,000 possible combinations of operating system software would comply with Section 1. If, as discussed above, almost any part of Windows that exposes an API meets the Section 22.x.ii definition—as it seems to—then the number of Windows variants would be much larger.

202. While a much smaller number of Windows variants would ever be offered by the marketplace, Section 1 would require Microsoft to design Windows so that *any* combination of "Microsoft Middleware Products" would "perform[ ] effectively and without degradation."

203. We have a very large engineering organization devoted to developing operating systems, but which is fully engaged in developing one major new operating system release every three years. There would be no way to build commercially acceptable operating systems consisting of countless combinations of components.

204. Fifth, Microsoft would be unable to test its operating systems effectively if it were obligated to test thousands of possible combinations. As I discussed in Section II.A.3 and II.A.4, testing is absolutely essential to ensuring product quality. In rough terms, about half the software development process for a single new version of Windows entails testing the product.

205. Absent thorough testing, there is no way that Microsoft could ensure that any "unbound" version of Windows that an OEM might create "performs effectively and without degradation." (That is the answer to witnesses for the non-settling States who say that Microsoft should test only a handful of the possible permutations of "unbound" versions of Windows.) Today it is difficult to test a single operating system release against

many combinations of other software and hardware products. If we expanded the test matrix to include all the various combinations required by Section 1, we would never finish the task.

206. Sixth, providing good product support for countless variations of an "unbound" Windows would be difficult and expensive, if not virtually impossible. Yet Section 3 of the NSPR would obligate Microsoft to do so (both "directly and indirectly"). Product support specialists are already challenged in determining the source of problems encountered by consumers, given the wide array of hardware and software products in use and the unforeseen problems that can arise from the interaction of such products. If Windows itself were no longer well-defined, Microsoft's product support specialists would find it virtually impossible to assist customers in resolving their problems.

b. Non-Settling States' Explanations of Code Removal

207. I have read the expert report, two depositions and direct and cross-examination testimony of Professor Appel. I have also read the deposition of the non-settling States Rule 30(b)(6) witness, Mr. Greene, and portions of depositions, direct and cross-examination testimony of the non-settling States' witnesses who testified about Section 1. The testimony of those witnesses tends to compound, rather than solve, the problems identified above.

208. Several of the witnesses appear to suggest that Microsoft could comply with Section 1 if it redesigned Windows in a "modular" fashion. But modularity is not the issue here. One can certainly identify various "modules" within Windows (and do so at varying levels of granularity). The code for Internet Explorer in Windows can be thought of as a module, as can the various components of Internet Explorer, such as the

MSHTML.DLL component referenced above, or even subparts of such a component. The problem presented by Section 1 is that various modules within Windows rely upon other modules, so that removing one module will impair other parts of the operating system.

209. In fact, one of the reasons why software is developed in a modular (or "componentized") fashion is to *facilitate* one module's use of another. In the liability phase, my colleague Jim Allchin explained that Microsoft undertook a major software development project in the mid-1990s to "componentize" Internet Explorer to facilitate use of its capabilities both by other parts of Windows and by third party developers via published APIs:

> . . . Microsoft set about the task in early 1995, before the first version of Windows 95 was even released, of tearing apart and then rebuilding Internet Explorer as a series of software components. Microsoft then "exposed" the functionality performed by these components in the form of hundreds of APIs. *This is a very important point*. Today the entire developer community benefits from Microsoft's inclusion of Internet support in Windows because all developers can call upon this built-in functionality in creating their own products.

(Direct Testimony of Jim Allchin at ¶ 85 (emphasis in original); *see also id.* at ¶¶ 48, 124-33.) Mr. Allchin went on to explain in detail how various parts of Windows rely on, and thus benefit from, six core components (or "modules") that make up Internet Explorer. (*See id.* ¶¶ 99-123.)

210. Mr. Greene, the non-settling States' Rule 30(b)(6) witness, appeared to testify that the "performs effectively and without degradation" language followed by the parenthetical actually means that some parts of the operating system—but not others—can be degraded when the binary code for a "Microsoft Middleware Product" is removed. As I understand it, Mr. Greene believes that Microsoft must ensure only that "basic" operating system functions continue to perform effectively and without degradation after the removal

of the software code. As an example, Mr. Greene testified that text in the HTML Help system is "basic" and should remain functional after Internet Explorer is removed, but that a more advanced means of communicating Help information—via audio clips (audio even multimedia help is available today in various Microsoft products)—is not "basic" and need not continue to function when the binary code it would rely upon (Windows Media Player) is removed from the operating system.

211. I do not see anything in Section 1 that draws a distinction between "basic" operating system functions that must continue to perform effectively and other functions that may be degraded. In any event, there is no such distinction as a matter of software engineering. A rule that only "basic" functions must continue to perform would be entirely subjective, particularly given the rapid pace of operating system development in which today's innovations are taken for granted tomorrow.

212. The non-settling States' technical expert, Professor Appel, devotes 3 ½ pages of his testimony to Section 1. (Direct Testimony of Dr. Andrew W. Appel at pp. 49-52.) His testimony is directed mainly at supporting his "opinion" that Windows "is most likely written in modular fashion." (*Id*. at ¶ 133.) As I explained above, however, whether Windows is written in a modular fashion says nothing about the feasibility of removing modules from Windows while expecting other parts of the operating system to continue to "perform[ ] effectively and without degradation."

213. I can say with confidence, based on twenty years' experience overseeing the development of successive versions of Windows, that (i) large parts of the operating system are "modular" in the sense in which Professor Appel is using that term

and other large parts are not; and (ii) removing significant parts of Windows—whether modular or not—will cause other parts of Windows that rely upon those parts to fail.

214. In a single sentence, Professor Appel appears to suggest that Windows could continue to perform effectively even after code is removed because an "interchangeable" module could be swapped into Windows in its place. (*See id.* at ¶ 131.) But this is one respect in which Section 1 is quite clear: it obligates Microsoft to ensure that Windows "performs effectively" upon the removal of Windows code, *without regard to whether other non-Microsoft software is installed to replace the missing functionality*.

215. Even if Section 1 did permit Microsoft to rely on a third party to supply missing functionality, Microsoft could never ensure that the operating system would "perform[ ] effectively" if arbitrary pieces of the operating system were removed in favor of non-Microsoft software. As one might expect, non-Microsoft software is designed differently from Microsoft software, performing different functions in different ways. For example, if one removed the binary code for Internet Explorer from Windows XP and substituted Netscape Navigator (or any other Web browser), many parts of Windows XP would fail. Internet Explorer exposes hundreds of well-defined APIs. Other parts of the operating system and numerous third-party applications call *those* APIs and expect *those* APIs to perform specific functions in a specific way. No other Web browser exposes the same APIs, and thus no other browser can perform the functions of Internet Explorer within Windows.

216. In a purely theoretical world, one could imagine developing modest software programs in such a way that any module could be swapped out in favor of a similar module developed by a third party. The replacement module would need to

conform identically to the interfaces expected by all of the modules with which it interacts. In the commercial world, it is hard to see what value such replace-ability would provide even if it could be achieved. For Netscape Navigator to suffice as a replacement for Internet Explorer, for example, developers at Netscape would have to devote enormous effort to matching the functions of Internet Explorer and enabling those functions to perform in precisely the same way as Internet Explorer. When they were done, they would have software that is nearly identical to Internet Explorer (a "clone"), providing little or nothing in the way of new value.

217. In addition, if Microsoft were obligated to allow ISVs to clone all the functions of all the "Microsoft Middleware Products" in Windows, Microsoft's ability to improve Windows would be hampered because the interfaces between modules would necessarily be "frozen" so that third parties could write to them. Given the large number of "Microsoft Middleware Products" in Windows under Section 22.x, the effect would be to freeze large parts of Windows.

218. On re-direct, Professor Appel expanded upon his brief direct testimony concerning how Microsoft could comply with Section 1. He testified that there were four ways in which Microsoft could do so. I believe that none would meet the obligations that Section 1 imposes. Professor Appel's four methods are as follows (*see* Trial Transcript, April 10, 2002 at 3208-3210.)

> a. *"One way is to simply let the Microsoft middleware product be removable."*

If the "binary code" for Microsoft Middleware Products is removed, everything else in Windows that benefits from that code will be degraded, violating Section 1.

> b. *"Another way is to let subcomponents of the Microsoft middleware products be removable. The States' remedy doesn't*

> *require that, but it permits that. And then in the case of, for example, MS HTML, the rendering engine the subcomponent of the browser, an OEM might choose to leave that component in even if they want to substitute a different browser, and then there's no chance of degradation of the functionality of other components that depend on that HTML rendering."*

This approach entails leaving important code *in* Windows, not removing it. Leaving code *in* Windows would certainly solve the "degradation" problem, but it would subject Microsoft to claims that it failed to permit code to be "removed." The HTML rendering engine is central to any Web browser and so one might expect it to be "removed" when the "binary code" for Microsoft's Web browsing software is removed. Moreover, Prof. Appel provides no guidance as to *which* subcomponents of any "Microsoft Middleware Product" could, in his view, be left behind, other than the HTML rendering engine in IE.

> c. *"Another option, as I have explained, is to take necessary fragments of functionality and embed them in other products, other than Microsoft middleware products, so they don't expose APIs."*

This approach also entails leaving important code *in* Windows, not removing it, in apparent violation of Section 1. Here, Professor Appel contemplates merely *moving* code to different places within Windows, putting it wherever it may be needed in Windows, and thereby enabling the code to continue to provide functions to other parts of the product. For example, we would put one copy of the HTML engine with the Help files, another with the mail client files, another with the file system files, another with the Web browser files, and so forth. Under this approach, "removing" the binary code for a Microsoft Middleware Product would actually entail leaving it—in fact, multiple copies of it—*in place and in use.* There is another problem: Professor Appel explained that under this approach Microsoft would be required to "not expose" APIs associated with the middleware code left in Windows. (*See* Trial Transcript, April 9, 2002 at 2989.) But those same interfaces would

be used by the other functions in Windows (such as Help using the HTML engine APIs), would have to be documented under Section 4 of the NSPR, and therefore *would be* exposed for use by any software developer.

> d. *"Another kind of way to comply is just to reduce the inherent commingling, or I should say interdependence between Microsoft middleware products."*

Here Professor Appel just assumes away the technical reality that makes Section 1 infeasible: any modern software product has modules of code that depend upon other modules of code. Professor Appel is proposing that Microsoft scrap a good part of our work in building the world's leading operating systems and design new operating systems that do not provide ISVs or consumers with the benefits of integration among components. Doing this would unquestionably result in "degraded" operating systems, systems that are much, much larger, slower, harder to debug, harder to secure and harder to fix when problems occur. Professor Appel's fourth removal "method" does not explain how Microsoft can make the code in Microsoft's actual operating systems "removable" without "degrading" the parts of Windows that rely upon that code.

219. Professor Appel and other witnesses for the non-settling States have testified that Windows XP Embedded proves that the "unbound" version of Windows can be created and that Microsoft has already done most of the engineering work needed to create such a version. This is wrong for several reasons which will be addressed in greater detail in the testimony of other Microsoft witnesses. However, I want to emphasize that Windows XP Embedded does not remotely resemble what the non-settling States' witnesses claim. Windows XP Embedded is a set of tools that can be used to create a customized embedded "runtime" from the binary code of Windows XP to run a single purpose device like an ATM or cash register. It is not a general purpose operating system.

220. It is true that Windows XP Embedded includes a database in which Microsoft has tried to define components comprised of software code in Windows XP that are associated with particular functionality, such as Internet Explorer or Windows Media Player. These component definitions describe which code must be *included* for a customized embedded runtime to perform specific functions that are selected using the Target Designer tool in Windows XP Embedded.

221. One might regard the definitions created by Microsoft for that specific purpose as being useful for defining what software code constitutes some of the components that Section 1 would require Microsoft to make "removable" from Windows, although the NSPR does not appear to vest discretion in Microsoft for making those determinations. In any event, defining what code constitutes such components is only the first step. Once defined, it is then necessary to identify all of the dependencies that a given component has to all of the other components of Windows XP. And that is where problems arise, because each component in Windows XP has dependencies on many other components. As a result, it is not possible to pull components out of Windows XP and expect everything else to keep working.

222. For example, the component needed just to run a particular sound card depends upon a great many other components of Windows, in some cases including all of the components involved in providing Web browsing functionality. Certain of these interdependencies may seem counterintuitive, but they reflect basic engineering efficiencies, and they make it extremely difficult to create a version of Windows from which various components could be removed without degrading the rest of the operating system.

223. The Windows XP Embedded component definitions therefore do not define what software code can be *excluded* and still have Windows XP function without degradation as a general purpose operating system. Most importantly, nothing in Windows XP Embedded identifies—much less tries to eliminate—all of the interdependencies in Windows that will cause the operating system to break if arbitrary blocks of software code are removed.

c. Consequences of Code Removal

224. Even if it were feasible to develop operating systems in compliance with Section 1's design requirements, the resulting products would provide little value to the marketplace.

225. Section 1 contemplates a world in which OEMs and anyone else that licenses 10,000 copies of Windows can remove the binary code of any "Microsoft Middleware Product" in Windows, while the balance of the operating system will continue to perform effectively. In other words, Section 1 contemplates a world in which there is not one version of Windows, but many. These different versions of Windows would each expose APIs.

226. If Windows were deconstructed and offered by OEMs and others in multiple variations, the primary value that Windows provides to the marketplace—a stable, consistent platform for software development—would be lost. Neither ISVs nor end users would have any assurance as to what functionality would be provided in any given variant of Windows.

227. For ISVs, the advent of multiple versions of each release of Windows would make it impossible to design products that rely on the presence of particular

functionality in the operating system. ISVs would thus be forced to include more functionality in their own products, making those products more time-consuming and expensive to develop and larger and more complicated generally, leading to more bugs and higher support costs. Many existing products would not run properly, or at all, on versions of Windows from which the binary code supporting necessary APIs had been removed. (*See* Demonstrative Exhibit 2.)

228. To provide a satisfactory customer experience, ISVs would have little choice but to develop new products designed to run only on specified new versions of Windows (*e.g.*, the Compaq Presario 2002 version), returning the industry to the fractured state from which MS-DOS freed it in the early 1980s.

229. Consumers would likely be confused and frustrated if OEMs and others were allowed to remove key parts of Windows and nonetheless sell the resulting software as "Windows" (as Section 2.b of the NSPR would expressly permit). Imagine the disappointment that would result if a consumer purchased a new Windows-based PC, only to discover that key new features (such as video instant messaging or Internet support) were missing and various applications would not work at all because of missing APIs. OEMs would have strong incentives to fragment Windows in this way because they would be paid by Microsoft competitors simply to remove code from Windows, or to remove code and replace it with competitors' code. Of course our competitors could contribute to such fragmentation directly by licensing 10,000 copies of Windows, which they would modify themselves and distribute to end users. If OEMs and others were permitted to produce myriad versions of Windows with varying feature and API sets, and identify them using the

Windows trademark in order to trade off the goodwill Microsoft has developed in that famous name, the Windows trademark would become meaningless.

230. If OEMs offered different versions of Windows, there would be (i) significantly less price competition among OEMs (because their products would be less interchangeable); (ii) greatly reduced interoperability between brands of PCs (because key interoperability technologies, such as protocols, might be different on different machines); (iii) greatly reduced interoperability between applications and PCs (due to missing APIs); (iv) increased development costs; (v) reduced development output and innovation; and (vi) considerable consumer confusion. These harms would likely lead to slower growth of the PC industry as a whole.

d. Pricing

231. Wholly apart from the many problems identified above, the pricing provisions of Section 1 would create a disincentive to developing improved versions of Windows. Rather than earn a return on our substantial investment in improving Windows, any improvements could result in a revenue *loss* to Microsoft. In fact, under the pricing formula set forth in Section 1, the price of Windows could be zero. That pricing regime also could not feasibly be implemented, for the reasons set forth below.

232. ***Disincentive to Improve Windows.*** Given the very broad definition of "Microsoft Middleware Product," nearly any new software development we do in Windows would likely be regarded as constituting a "Microsoft Middleware Product" under the NSPR. Yet the result of any effort to develop a new capability would result in a *reduction* in the price of Windows. Under Section 1, any OEM or anyone else that licenses 10,000 copies of Windows could "remove" the "Microsoft Middleware Product," entitling

it to a corresponding price reduction. Since, in the non-settling States' view, Windows consists largely of "Microsoft Middleware Products" (a bare minimum of ten), many OEMs and other licensees would "remove" enough of the operating system to earn the 25% price reduction called out in Section 1.

233. OEMs may well be paid to remove parts of the operating system by Microsoft competitors who want their "middleware" to be featured exclusively. Those payments together with the resulting Windows price discount would provide OEMs with an even stronger economic incentive to remove Windows code to earn short term profits, even though consumers would be disadvantaged.

234. At a minimum, every OEM would have a strong incentive to "remove" the "Browser" software from Windows, earn the price reduction that would flow from our roughly $100 million in annual development costs for that software (very roughly, 15% of the cost of developing a desktop version of Windows), then add back a free version of the exact same "Browser" software made available under the compulsory, royalty-free source code licenses for Internet Explorer provided under Section 12. Over the ten year life of the NSPR, OEMs' perfectly rational decision to swap out then add back identical "Browser" code would result in revenue loss to Microsoft of roughly $10 billion. Under such a pricing scheme, why would any rational business enterprise in Microsoft's position continue to invest in Web browser innovation, whether as part of Windows or separately from Windows?

235. Section 1 would also create a strong disincentive for Microsoft to continue making components of Windows available separately from the operating system at a positive price. Whenever we offered consumers the option of acquiring some part of the

operating system separately from a full operating system release, any OEM or other third party licensee could take a dollar-for-dollar price reduction on the price of Windows, without the 25% cap otherwise provided for in Section 1. By "removing" from Windows just two "Microsoft Middleware Products" that Microsoft also licenses separately from Windows for $35 or more, an OEM or other third party could reduce the price paid for Windows to zero. (Typical OEM pricing today for the consumer version of Windows XP is less than $70.)

236. In view of the vagueness and ambiguity in the NSPR's definition of "Microsoft Middleware Product"—and the OEMs' incentive to reduce costs wherever possible—Microsoft would be constantly subjected to claims that various Microsoft applications constitute or include "Microsoft Middleware Products" that also appear in Windows, entitling the OEMs to remove corresponding software from Windows and take large price reductions. For example, Windows XP includes rudimentary tools to enable consumers to view and manipulate digital photographs (an increasingly popular file type). Microsoft also offers an application that provides a far more complete set of tools for working with digital photographs, called Microsoft PictureIt!, at an estimated retail price of $34.95 (typical street price is lower). Since "digital imaging software" is explicitly included in Section 22.x.i as a form of "Microsoft Middleware Product," we would likely face claims under Section 1 that OEMs should be free to "remove" the digital imaging software in Windows and thereby earn a price reduction of $34.95 or some significant part of that.

237. If the NSPR had been in effect in the early 1990s, it would have made no sense for Microsoft to develop Windows 95, one of the most successful software

products of all time. Windows 95 was a new operating system that provided a wide range of improvements over Microsoft's existing offerings, including improved graphical user interface capabilities (which ISVs could use in developing their own products). Prior to the release of Windows 95, Microsoft offered MS-DOS and Windows 3.x as two separate products that, when used together, provided graphical user interface capabilities. Windows 3.x provided a broad range of APIs and thus would likely have been regarded as a "Microsoft Middleware Product." Under our price guidelines at the time, Windows 3.1 was available to OEMs at a royalty between $21 to $36, and to consumers for a suggested retail price of roughly $150 (or a price to distributors of $80). (Street prices were lower, and various discounts were available from all the numbers cited here.) Windows 95 was initially made available to OEMs at a price of roughly $42 to $75. Under Section 1, any OEM who wanted to take the many benefits of Windows 95, but pay much less for it, could modify the operating system in some way to allow it to say that it had "removed" Microsoft's new graphical uses interface (a "Microsoft Middleware Product") and thereby claim a price reduction of roughly $150, $80 or $21-$36 (Section 1 is ambiguous as to which pricing should be used), possibly reducing the price of Windows 95 to zero.

238. Section 1's pricing regime is not consonant with the basic economics of the software business. Given the low marginal cost of distributing software, software programs typically are priced not as a function of development costs (as Section 1 attempts to do), but rather as a function of (i) the value they provide and (ii) the likely size of the customer base over which costs can be amortized and revenues generated to fund future development. (That is why highly specialized products tend to be priced higher than more widely used products.) Section 1's pricing regime turns software economics on its head by

making the price of Windows lower (for any OEM who elects to "remove" software) as Microsoft adds more value to the product.

239. Given the pricing provisions of Section 1, the only economically sensible plan for Microsoft would be to develop all future platform innovations as products that are made available *only* separately from Windows. In other words, Windows would be frozen for the duration of the NSPR at the particular level of functionality that it provides today. Within a few years, as competing platforms raced ahead by adding new capabilities, Windows would become obsolete. And even before then, sales of Windows would decline rapidly. Since software never wears out, consumers would have little incentive to purchase a new version of Windows if the new version provided basically the same features as the old version.

240. ***Feasibility.*** Microsoft does not track "development costs" along the artificial lines of "Microsoft Middleware Products"—an input to the price formula set forth in Section 1. Given Section 22.x's vague definition of "Microsoft Middleware Product," and the fact that software in Windows is used for multiple purposes by multiple parts of the operating system, no accurate accounting of those development costs would even be possible because we would not know which software to include in the costs for each "Microsoft Middleware Product."

241. If the definitions were modified to make clear what software constitutes each "Microsoft Middleware Product," we could come up with reasonable estimates for the cost to write the code for a few of the major portions of Windows that the NSPR specifically defines to be "Microsoft Middleware Products," such as Internet Explorer. We do not, however, attempt to track costs associated with the many smaller parts

of Windows that the NSPR also deems to be middleware. Even in the few cases where we could establish reasonable estimates, there would be no sensible way of accounting for all the testing we do on the product as a whole—which accounts for roughly half the cost of developing a new release of Windows.

242. Another problem is presented by the lack of clear direction in the NSPR as to what it means to "remove" "Microsoft Middleware Products" from Windows. (*See* Section III.B.1.a.) As discussed above, that apparently could mean removing "binary code" for "Microsoft Middleware Products," as Section 1 says, or leaving some or all of the code in. Drawing upon that lack of clarity, OEMs would have an economic incentive to modify Windows in any way they see fit consistent with Section 2.c, claim they have "removed" a "Microsoft Middleware Product," and demand a corresponding price reduction. In many cases, OEMs will leave most of the relevant "binary code" in Windows (since the code is useful), removing just a little while claiming that they have "removed" the "Microsoft Middleware Product" under Section 1 and are therefore entitled to a price reduction. Given the lack of clarity in Section 1, there will be no clear way to determine whether the OEMs have really "removed" any "Microsoft Middleware Product" for Section 1 purposes. Since price is very important, this section of the NSPR is likely to generate a lot of conflict.

2. Section 2

243. Section 2 regulates three aspects of Microsoft's business: it requires Microsoft to license its Windows technology to OEMs and others that are not similarly situated on uniform terms (Section 2.a); it requires Microsoft to treat OEMs and others that are not similarly situated "equally" with respect to many things, including development and

co-marketing work (Section 2.b); and it requires Microsoft to allow OEMs and others to modify Windows more or less any way they like, including removing software code from the products while still promoting the resulting software as "Windows" (Section 2). The effect of Section 2 would be to exert a strong upward pressure on OEM pricing for Windows, greatly discourage collaborative development and marketing efforts between Microsoft and OEMs, and greatly devalue Windows by taking away the consistency it provides today across PCs from different OEMs.

a. Section 2.a

244. Section 2.a would obligate Microsoft to license Windows on uniform terms and conditions. The "uniform terms and conditions" requirement of Section 2.a extends not only to the twenty largest OEMs (so-called "Covered OEMs"), but also to *anyone that licenses just 10,000 copies "for commercial purposes"* and thereby becomes a "Third Party Licensee." (Section 22.oo.) As a result, Section 2.a applies to roughly 150 of Microsoft's OEM customers and, by the express terms of Section 22.oo, to companies that are *not* OEMs, such as "systems integrators and value-added resellers," large corporate customers, and, of course, competitors such as AOL Time Warner.

245. It does not make business sense to license software products to licensees in different distribution channels—and even end customers (large corporations) on "uniform terms and conditions," including a single price discount schedule. OEMs, system integrators, value-added resellers and others are in very different businesses, and their licenses must reflect those differences. Like other software companies, Microsoft finds it efficient to license and price its software to reflect differences among distribution channels.

246. The essence of an OEM license is an intellectual property grant from Microsoft to the OEM to pre-install a copy of Windows on a new PC for sale to an end-user. Microsoft offers Windows to OEMs at lower prices than it does to distributors in other channels to reflect the value that OEMs provide to Microsoft by building, promoting and supporting PCs that run on Windows. Microsoft's OEM pricing thus reflects (a) the integration and testing work OEMs do with their hardware, (b) the simplification for the consumer of acquiring Windows pre-installed on a new PC, (c) the product support costs OEMs bear, and other factors. Distributors in other channels do not provide such useful services.

247. A competitor such as AOL Time Warner that licenses Windows for sale to end users will not provide the type of services that OEMs provide today. Indeed, it is unlikely that AOL would do anything to promote the value of the Windows platform. AOL's business interest lies in obtaining rights to as much of Microsoft's technology as it can, as cheaply as it can, and then in using that technology to compete against Microsoft in an effort to make Windows users customers of AOL's own software and services over time.

248. The NSPR establishes a structure that would essentially defeat the concept of a price discount curve. Since any Third Party Licensee is free to redistribute Windows to anyone else, it would make economic sense for smaller OEMs, VARs and so forth to aggregate their volumes in licensing Windows from Microsoft so they would qualify for the largest possible discount.

249. Section 2.a would also prohibit Microsoft from offering OEMs price *discounts* on Windows royalties pursuant to Market Development Programs ("MDPs") to encourage them to undertake various activities that serve to promote the Windows-based

PCs. These MDP discounts are a way for Microsoft to share the cost of engineering and promotional activities that benefit both OEMs and Windows. By offering OEMs financial incentives to undertake such activities, Microsoft is able to obtain some of the efficiencies that accrue to vertically-integrated firms such as Apple that manufacture their own PCs, rather than license their operating systems broadly to other PC manufacturers. These discounts benefit consumers by making new hardware technology more broadly distributed so that ISVs will write new applications to support those technologies, making new computers more attractive.

250. For example, Microsoft might offer an OEM a discount for ensuring that its PCs include support for new methods of connecting peripherals to PCs, such as USB and "Plug and Play" technology, which enables Windows to recognize a new device automatically and set it up with little or no user intervention. Microsoft currently offers OEMs a discount for ensuring that their PCs distributed with Windows boot up within a defined period of time. If an OEM believes that new connection technologies are not important, or believes that the engineering cost needed to implement the new technologies is greater than the associated Windows price reduction, the OEM need not undertake that activity.

251. Beyond price, Microsoft's Windows license agreements include various terms that are relevant only to particular distribution channels. It would not be feasible to require that Microsoft impose identical terms on licensees engaged in different lines of business. An OEM such as Dell that installs Windows on more than 15,000,000 PCs per year shipped worldwide has different needs when licensing Windows than a small

value-added reseller, which may have a dozen employees assisting corporate customers in acquiring and maintaining complete computing solutions.

252. By requiring that Microsoft license Windows to OEMs and non-OEMs pursuant to uniform terms and conditions, including price, and by flatly prohibiting MDP discounts, Section 2.a would create strong upward pressure on OEM pricing and reduce collaboration between Microsoft and OEMs that build on Windows technology.

b. Section 2.b

253. Section 2.b would be nearly impossible to implement in the real world. Section 2.b requires that Microsoft treat "equal[ly]" entities that are not "equal" at all, and it apparently imposes such a requirement with respect to *any* Microsoft technology, not just Windows, although the language is ambiguous on this point.

254. Under Section 2.b Microsoft cannot enter into many kinds of beneficial business relationships with a large OEM such as Compaq or Dell without also offering to enter into the same relationships with the 150th largest OEM (shipping only 10,000 Windows-based PCs per year). Moreover, Microsoft would have to offer "equal" business relationships to anyone else that licenses 10,000 copies, even if they are not an OEM and have no interest in promoting the Windows platform.

255. Section 2.b could effectively shut down engineering collaboration between Microsoft and large OEMs. Some OEMs, such as Compaq and Hewlett-Packard, have large engineering organizations. Other OEMs are essentially marketing organizations, making no investment in engineering and even outsourcing the manufacturing of the PCs they sell. Needless to say, Microsoft devotes engineering resources to working with other engineering organizations, not with marketing organizations. Yet Microsoft would violate

Section 2.b if it provided any "technical information" (the essence of an engineering discussion) to one OEM and not to another. In fact, any time Microsoft engaged in long-term planning with a key OEM relating to future technologies, Microsoft would be obligated to engage immediately in similar discussions with 150 other OEMs—and direct competitors such as Sun, Palm, and AOL if they licensed 10,000 copies for commercial use. That simply is not practical.

256. Similarly, Section 2.b would make it impractical for Microsoft to enter into routine and beneficial co-marketing agreements with OEMs. For example, an OEM building a PC intended to appeal to students might propose to work with Microsoft on promotional activities, such as advertising or joint sales calls to campus administrators. Microsoft might find such joint promotion appealing to promote Windows to students. But Section 2.b would proscribe such co-marketing activity unless Microsoft was prepared to devote "equal" resources—money and time—to all third parties that license 10,000 copies of Windows, even if they had no particular commitment to the student market. That would not make business sense. Faced with the inability to use our resources efficiently, we would have no practical alternative but to cease engaging in co-marketing with OEMs.

257. Section 2.b is striking in that it would obligate Microsoft to enter into relationships that are "equal," in a long list of specified respects, both with entities that advance the Windows platform—OEMs building PCs based on Windows—and with entities such as Sun, Palm and AOL whose business interests are to the contrary. For example, Section 2.b would require Microsoft to provide highly sensitive "information about future plans" to Sun and AOL as soon as Microsoft had a discussion with any OEM on the subject. Here again, Section 2.b would make it essentially impossible for Microsoft

to work closely with its OEM customers. Like any technology company, we could not stay in business for long if we had to share information about future product plans with direct competitors.

258. The part of Section 2.b that refers to "bona fide joint development efforts" does not solve the problems identified here. Microsoft's relationships with Covered OEMs and Third Party Licensees generally are not joint development efforts.

259. By requiring that Microsoft treat "equally" anyone that licenses 10,000 copies, regardless of their ability to contribute to the Windows platform, the line of business in which they are engaged, or even their size, Section 2.b would greatly reduce Microsoft's ability to work constructively with a wide range of third parties, to the detriment of Microsoft, third parties, and consumers.

c. Section 2.c

260. Section 2.c is similar to Section 1 in that it is another way of enabling third parties, including competitors, to deconstruct Windows. What is striking about Section 2.c, however, is that it is much broader than Section 1. Section 2.c essentially lets any OEM or Third-Party Licensee modify Windows any way they like—and still promote the resulting software to consumers as "Windows."

261. Like Section 1, Section 2.c would deprive Microsoft of the ability to design a product and ensure that it would reach consumers in an unadulterated fashion. Section 2.c largely transfers control over Windows from Microsoft, which creates it, to anyone that licenses 10,000 copies or more.

262. Section 2.c would permit OEMs and others to modify Windows beyond recognition. Under Section 2.c, OEMs and others could remove any software code

for a "Microsoft Middleware Product" or, if they prefer, hide the code, making it hard or impossible for customers to find. They could also make non-Microsoft "Middleware" the "Default Middleware"—a term that is ambiguous, as I explain below. And they could modify nearly any aspect of the user interface of Windows or, if they like, simply remove altogether the "icons, folders, links, start menu entries, smart folder application or service menu entries, favorites, or other means of presenting Microsoft products, services, features, or technologies" from the Windows desktop. OEMs and others also could simply "display another user interface"—even though the Windows user interface is the most readily identifiable part of the product to consumers.

263. To be clear, Section 2.c does not appear to be limited to removing "Microsoft Middleware Products" from Windows. Section 2.c apparently lets OEMs and others remove the "means of presenting" just about anything in Windows, whether or not it is related in any way to a "Microsoft Middleware Product."

264. Microsoft seeks to promote sales of Windows and Windows-based PCs and software by developing new capabilities in Windows and promoting them to consumers. In the case of Windows XP, we promoted new capabilities such as the ability to (i) connect a digital camera and easily upload images to the PC; (ii) communicate with friends and family with full-motion video (not just text messages); (iii) ask a friend, family member or product support professional to take control of your PC from afar to fix something for you (remote assistance); (iv) edit a document in Word or non-Microsoft word processing software and let users on other PCs see your changes instantly; (v) post digital photographs easily to a Web site (operated by Microsoft or third parties); (vi) send digital photographs to a film processor; (vii) obtain updates, such as security fixes, to Windows

while you sleep; and (viii) search for information more easily on your PC or on the Internet, and more. *Under Section 2.c, OEMs and others would be free to "remove" the means of presenting all these Microsoft "features or technologies."*

265. If Section 2.c were entered, OEMs would likely accept payments from third parties to change Windows in various ways. For instance, AOL would likely pay OEMs to remove all or nearly all the features listed in the preceding paragraph because those features compete with services offered by AOL—even though AOL does not offer capabilities such as video instant messaging, remote assistance, or a choice of photo-finishing services.

266. Alternatively, as an AOL executive suggested it would like to do, AOL could just license Windows directly from Microsoft, obtaining the benefit of Microsoft's investment in developing Windows, and develop an "AOL Windows"—a product that would be directed solely at advancing the business interests of a single firm, AOL, rather than the broad range of ISVs and OEMs that benefit from Windows today (and that are essential to Microsoft's successful business model). (*See* Direct Testimony of John Borthwick at ¶¶ 33-34 and accompanying CD-ROM, PX 1709.)

267. Section 2.c would be bad for consumers (and thus for Microsoft). First, Section 2.c includes no safeguards to ensure product quality. Windows is a complex product. Once OEMs and others start modifying Windows, quality is certain to suffer. Microsoft would have no way to assure customers that Windows will function properly when modified by an OEM or third party. And since Microsoft makes Windows, consumers would likely lay the blame for any failings at our feet.

268. Second, Section 2.c is bereft of consumer protection safeguards. Section 2.c says that OEMs can remove or modify nearly anything they like in Windows, yet it guarantees OEMs "permission to display trademarks and logos," such as the Windows trademark. Consumers will be frustrated and disappointed—if not outright deceived—if they purchase a new PC bearing the Windows XP trademark, only to get home and find that one Windows XP feature or another is missing, or that the computer does not work well or does not run a favorite program because the version of Windows XP installed on the computer has been modified.

269. Third, Section 2.c would make it more difficult for consumers to walk up to any PC running Windows and feel comfortable that they know how to use it. With widely varying interfaces and feature sets, consumers would have to devote considerable effort to learning how to use PCs from different manufacturers.

270. It would be in the short-term economic interest of each OEM to accept payments to remove features from Windows or otherwise modify the product in ways that reduced its value to the marketplace as a whole. Each OEM is focused (quite rationally) on its own economic interests, rather than the interests of the larger PC ecosystem. In the early days of the PC industry, for example, OEMs were content to offer their own operating systems, hoping to generate developer interest in creating applications that ran only on their PCs. That is why it took a third party (Microsoft), not any OEM or group of OEMs, to develop a common operating system platform that the PC industry could rally around.

271. Microsoft's interests in developing and promoting Windows are a closer proxy for the interests of the PC ecosystem because Windows is the key technology

that provides compatibility across a broad range of hardware and software products from thousands of companies. We know that improvements to Windows that enable OEMs to build better PCs, and ISVs to build better applications, will lead to increased sales of those products and thus increased sales of Windows. Therefore, Microsoft works hard to ensure the integrity of the Windows platform. For an OEM, however, it might make sense economically to remove features from Windows in exchange for cash from a Microsoft competitor such as AOL. Similarly, OEMs might find it attractive to accept payments to put advertising in the Windows user interface; Microsoft has elected not to do so in order to maintain the best user experience for all Windows customers.

272. Over the long-term, modifications to Windows by individual OEMs acting in their short-term self interest would present a classic tragedy of the commons problem. Just as a lake that is fished too heavily soon will support no one, the PC ecosystem as a whole will suffer if the stability and consistency of Windows is not maintained, for the reasons I discussed above. When PCs become less reliable because the quality of Windows has been compromised, when consumers must undergo retraining to operate different brands of PCs because of differences in their user interfaces, when applications written for one version of Windows will not run on another version, the entire PC ecosystem will suffer.

3. Section 3

273. Section 3 would require Microsoft to continue to license old versions of Windows for five years following the release of any major new version. Microsoft releases a major new version of Windows approximately once every three years. Section 3 thus would require Microsoft to license every major new operating system it releases for

eight years—without changing any of the licensing terms. Section 3 also prescribes the price for such operating systems: no more than the lowest royalty rate each licensee paid before the term began. While licensing old operating systems at old prices and on old terms and conditions, Microsoft must assume the burden of "support[ing]" the operating systems both "directly and indirectly."

274. Section 3 would create three primary problems: (a) it is yet another method of promoting the marketing of multiple, fragmented versions of Windows; (b) it would lead to considerable consumer confusion; and (c) it would slow Microsoft's efforts to promote the PC ecosystem by moving the marketplace to improved versions of Windows that provide new and important capabilities.

a. Proliferation of Windows Variations

275. As discussed above, Section 1 would create opportunities for OEMs and others to ship Windows in any of 4,096 variations within six months after the NSPR went into effect. Section 2 contemplates many more versions of Windows as OEMs and others would be free to modify Windows in still more ways. Section 3 provides that any of these thousands of versions will be available in the marketplace for five years after Microsoft has released a successor operating system. Accordingly, eight years from today, we would potentially have thousands of versions of Windows XP in the marketplace, thousands of versions of Windows XP+2 (the successor to Windows XP, code-named Longhorn), thousands of versions of Windows XP+4, etc.

276. Even if the Windows platform were fragmented by the development and widespread licensing of just ten variant versions where significant blocks of code were removed, the PC ecosystem would suffer the harms I discussed in Section III.A above.

Among those harms would be an inability to provide good product support for variations of Windows we did not create—even though Section 3 expressly obligates us to provide such support.

b. Consumer Confusion

277. The proliferation of countless variations of Windows, some up to eight years old, would likely be confusing to consumers. Operating system technology advances rapidly over eight years, and hardware and software built for new versions of Windows will not work correctly, or at all, with older versions. For example, new printers today often connect to a PC via a USB port. Less than eight years ago, PCs did not ship with USB support, and neither did Windows. Given the complexity of computing, consumers may have a hard time ascertaining the significance of purchasing a PC with one variation of Windows developed over the preceding eight years as opposed to another.

c. Slowed Growth of the PC Ecosystem

278. Microsoft puts a lot of effort into promoting new releases of Windows because widespread adoption of new releases promotes the growth of the entire PC ecosystem (which benefits Windows). ISVs benefit from new platform capabilities in each new operating system release, which benefits consumers, which causes them to buy new PCs to run new versions of Windows, which in turn spurs sales of applications, and so on. Obsolete versions of Windows are a drag on the ecosystem. Lacking the latest advances, they often cannot run the latest software, connect to the latest peripherals and so forth.

279. If Microsoft were unable to phase out obsolete operating systems in a timely way, the PC ecosystem would likely move more slowly to new Windows releases,

retarding the overall advancement of the platform. The requirement that Microsoft continue to license and support old operating systems also would undercut Microsoft's efforts to make computing more secure and reliable via improvements in new releases of the Windows operating system.

d. Security and Patents

280. There are a variety of reasons why it might be important over the next ten years for Microsoft to be able to withdraw an old operating system from the market. For example, a serious security flaw might be discovered that cannot be adequately addressed with a software patch. Or we might learn that an old operating system violates a valid and enforceable patent to which Microsoft cannot obtain a license. Yet Microsoft nevertheless would be required to distribute the old operating system under Section 3.

4. Section 4

281. Section 4 would expropriate vast amounts of Microsoft's intellectual property, requiring Microsoft to provide all of its competitors (and everyone else in the industry) with proprietary and confidential information concerning the inner workings of "Microsoft Platform Software." The NSPR defines "Microsoft Platform Software" to include not only Windows running on PCs, but also *server versions of Windows, Microsoft Office, and any other Microsoft software, running on any kind of device, that provides functionality similar to "Middleware" offered by a competitor*. (Sections 4, 22.w, 22.x, 22.y, 22.rr.)

282. Since intellectual property is vastly less valuable if held by two entities or more, and since Section 15 explicitly states that Microsoft will receive no

compensation for its intellectual property, the NSPR would essentially effect a huge divestiture of Microsoft's key assets.

283. The title of Section 4.a—"Interoperability Disclosure"—is not a fair description of Section 4. Windows already supports tens of thousands of compatible software and hardware products. Throughout government and private industry, server software from Sun, Oracle, IBM, Hewlett-Packard, Linux vendors and others interoperate today with PCs running Windows, and each successive release of Windows makes our operating systems even more interoperable. Section 4 is about something else altogether: enabling Microsoft's competitors to have a free ride on Microsoft's platform R&D, both work we have done in the past and ongoing R&D over the next ten years. Section 4 is directed at giving Microsoft's platform technology to its competitors so that they can use it to improve their own competing products, without regard to any interoperability with Windows.

284. Section 4 explicitly states that competitors could use Microsoft's platform technology for interoperation between *non-Microsoft* platform software and *non-Microsoft* applications that were written to run on Windows. As a result, Section 4 enables Microsoft's competitors to use the information they obtain from Microsoft to build software solutions where no Microsoft software is involved at all.

285. Section 4 presents three sets of problems, discussed below. First, it would provide Microsoft's competitors with the equivalent of the blueprints to Windows (both client and server)—and actual Microsoft software built to those blueprints. With free access to Microsoft technology, it would be relatively easy and inexpensive for them to develop a functional equivalent to Windows, *i.e.*, a Windows clone. Rather than develop

their own directory software, for example, a competing platform vendor could just provide its own version of Microsoft's Active Directory. Microsoft would have little reason to continue to invest in the Windows platform if it were required to disclose its intellectual property to its competitors without an opportunity to commercialize it (as required by the "Timely Manner" rules).

286. Second, Section 4 would present immediate compliance issues because the key terms are at once extremely broad and yet ambiguous in important respects. If we were required to comply with Section 4, we would not know where to begin. Nearly any interaction among any of the tens of millions of lines of code in Windows might be deemed to constitute an "API" or other information that must be disclosed under Section 4.

287. Third, assuming compliance were feasible, Section 4 would benefit Microsoft's platform competitors greatly, while harming consumers, the PC ecosystem and Microsoft. If Section 4 were put into effect (a) the Windows platform would fragment, with all the harms that entails; (b) applications written to internal interfaces that were not designed to support software external to the operating system would malfunction, making computers less reliable and harder to use; (c) Microsoft's ability to develop new versions of Windows, which requires changes to internal interfaces, would be greatly constrained; (d) PCs would be less secure; and (e) Microsoft would have no reason to continue to invest in platform software.

288. I discuss these points, in turn, below.

a. Asset Expropriation to Enable Cloning

289. Section 4 would provide Microsoft's competitors with vast amounts of information, even source code, that would greatly assist them in cloning Microsoft's

most important asset, Windows. It is, of course, far easier simply to mimic all the functionality of a rival's product rather than to create something new. Once provided with the equivalent of the blueprints for Windows, competitors such as Sun would have little trouble writing their own implementation of everything valuable that Windows provides today, including the capabilities it provides to developers via APIs. (*See* Demonstrative Exhibit 3.)

290. Competitors could develop products that implement Microsoft's Windows technology at very low cost since they would have access to all of Microsoft's R&D investment for free. Bearing little R&D cost, competitors could effectively render Windows irrelevant by licensing their implementations of Microsoft's technology at zero or low cost. And, our competitors could add proprietary improvements to Windows for ten years, all protected by any applicable intellectual property rights, while Microsoft would be largely prevented from making its own improvements unless they were shared with all competitors.

291. The non-settling States clarified that the objective of Section 4 is to enable such cloning when they revised their remedy proposals to state explicitly that Microsoft must disclose its otherwise confidential information "for the purpose of enabling non-Microsoft Platform Software . . .to Interoperate with . . . *applications for Microsoft Platform Software*." The only way a non-Microsoft operating system can "Interoperate" with the tens of thousands of applications written by third parties to run on Windows is if the operating system implements (*i.e.*, clones) all of the Windows APIs.

292. The full breadth of the asset expropriation contemplated by Section 4 is made clear by (i) the product categories to which the required disclosures relate

(essentially, all product categories); (ii) the scope of the defined terms; and (iii) the provision that would allow Microsoft's competitors to "study, interrogate and interact with" the source code for all "Microsoft Platform Software."

293. ***Product Categories.*** The disclosures that would be required by Section 4 extend far beyond promoting the development of non-Microsoft middleware to run on Windows or even promoting interoperability between non-Microsoft server operating systems and PCs running Windows. In fact, as noted, Section 4 is not limited to interoperability with Windows Operating System Products at all.

294. The scope of the intellectual property that Microsoft must disclose is correspondingly broad. Section 4.a.i requires disclosures relating to the interoperation of Microsoft applications and Windows—we already disclose so much information of this type that tens of thousands of applications run on Windows. Under the NSPR, all of our good work in this area would be subject to contempt risk.

295. Section 4.a.ii requires Microsoft to disclose interfaces that are internal to Windows and thus are not designed for use by external software.

296. Section 4.a.iii requires disclosures relating to interoperability between any Microsoft software on *any* device (handhelds, set-top boxes, etc.) and any "Microsoft Platform Software" running on any other device. Given the definitions of "Microsoft Platform Software" and "Microsoft Middleware Products," Section 4.a.iii appears to require Microsoft to disclose information relating to the interaction between *any two pieces of Microsoft software interacting across any two computing devices*. As noted above, once Microsoft has provided its technology to its rivals, they are free to use it in their own platform software to provide platform functionality to other non-Microsoft

software. Section 4 thus is not limited to interoperability with Microsoft operating systems for Intel x86 chips in any way.

297. ***Defined Terms.*** To understand the breadth of the disclosures required by Section 4, it is important to focus on the key defined terms. These terms are defined far more broadly in the NSPR than they are typically used in the industry. Most importantly, the definitions make clear that Microsoft is required to provide its competitors not only with interface information, but also with software that provides the functionality beneath the interfaces, *i.e.*, the features and functionality through which Microsoft competes with its rivals.

298. In industry parlance, an application programming interface is just that—an interface that applications use to call on services in an operating system. Along with the interface, Microsoft publishes documentation that explains how to use the interface. This documentation specifies what information the software program must provide to call on the API as well as what information, if any, the API may return to the program. Documentation for an API generally does not include information concerning how the system services beneath the API perform their functions. Rather, the beauty of exposing system services via APIs is that developers need not delve down deeper into the workings of the underlying technology.

299. Sections 22.c and 22.nn of the NSPR define the terms "API" and "Technical Information" far more broadly. Section 22.c defines "API" to include not only "interfaces," but also a long list of things such as "service provider interfaces, file formats, data structures, Component Object Model specifications and interfaces, registry settings, global unique identifiers ("GUIDs") and protocols." The definition then repeats that the

term “API” is not limited to “interfaces,” but extends to “methods, routines and protocols” that anyone might want to use for a very wide range of purposes.

300. In other words, the Section 22.c definition of “API” entails disclosure of how Microsoft provides the *functionality underneath* the API, yet the very purpose of an API is to provide an abstracted concept. APIs can become less useful when information is published about the functionality underneath them because developers may then rely upon details of the implementation that are likely to change over time, rendering the API and software that relies upon it “fragile.” Absent the kind of disclosure that Section 4 would require, a platform vendor such as Microsoft is free to improve its APIs by changing the underlying implementation.

301. The already broad term “API” is broadened still further by the definition of “Technical Information” in Section 22.nn. “Technical Information” includes “all information” regarding APIs and Communications Interfaces that a competent software developer would require to “Interoperate effectively.” Given the complexity of software, and the breadth of the defined term “Interoperate,” that definition would give competitors room to argue that nearly anything in Windows must be disclosed as “Technical Information” on the ground that they “require” it to “Interoperate effectively.” Section 22.nn also makes clear that “Technical Information” includes “but is not limited to” a long list of things such as “reference implementations, communications protocols, file formats, data formats, syntaxes and grammars, data structure definitions and layouts, error codes, memory allocation and de-allocation conventions, threading and synchronization conventions, functional specification and descriptions, encryption algorithms and key

exchange mechanisms for data translation, reformatting, registry settings and field contents."

302. It is hard to know what aspect of Windows Microsoft would *not* be required to disclose under Section 4. For example, Section 4 would require Microsoft to provide "all information" concerning the software interactions specified in Section 4.a.(i)(iii), "includ[ing] but not limited to reference implementations." (Section 22.nn.) A reference implementation means actual source code showing how to provide the functionality of a part of Windows or other Microsoft software. No such "reference implementations" exist for the vast majority of interactions specified in Section 4 other than the source code for Windows itself. Since Section 4 nonetheless appears to require that "reference implementations" be provided, Microsoft apparently would be required to provide its competitors (and everyone else in the industry) with the actual source code (*i.e.*, Microsoft's implementation) for Windows and other Microsoft software. In the alternative, we would be required to develop *new* software that performs the relevant functions of Windows (and other Microsoft software) and provide that to our competitors.

303. Professor Appel testified that Section 4 would not obligate Microsoft to create reference implementations that do not exist. (Trial Transcript, April 10, 2002 at 3183 – 3186.) His viewpoint, however, is not reflected in the language of Section 4 because the source code for Windows could be deemed to be a reference implementation that exists.

304. The definition of the term "Communications Interfaces" is similarly broad—and confusingly duplicative of the term "API." (Section 22.l.) The essence of both

terms appears to be “interfaces” and “protocols” that enable a wide range of software to “Interoperate” with Microsoft Platform Software.

305. The term “Interoperate” is defined not to mean “interoperate” as that term is used in the industry, but rather the ability to create the functional equivalent (a clone) of another firm’s software. In everyday usage, various software and hardware products interoperate with one another in various ways. Under Section 22.q, however, the term “Interoperate” is defined to mean “the ability of two products to effectively access, utilize and/or support the full features and functionality of another.” Given the complexity of modern commercial software, such “interoperability” would generally be achievable only if software products from various firms were nearly identical to one another, all adhering rigorously to a set of very well-defined specifications from which no variation was permitted. In the marketplace, however, commercial software vendors seek to compete on the basis of varying features and functionality, which means that products from different firms are not perfectly interchangeable with one another.

306. Further discussion of why the definition of “Interoperate” is not suitable for real software systems will be provided in the testimony of Professor Stuart E. Madnick.

307. ***Free Access to Microsoft Source Code.*** Like most other commercial software vendors, Microsoft generally seeks to limit access to its source code. Source code reveals product innovations. For example, a competitor who is free to review Microsoft’s source code (as Section 4.c permits under the misleading heading “Compliance”) will see the architecture, data structures, algorithms and other key aspects of the relevant Microsoft

product. That will make it much easier to copy Microsoft's innovations, which is why commercial software vendors generally do not provide source code to rivals.

308. When commercial vendors do provide source code to third parties (as Microsoft does in various circumstances), the relevant source code licenses usually place appropriate limits on who can see the source code and the uses to which it can be put. Given the scope of the defined terms in Section 4, however, Microsoft's rivals could seemingly use innovations they saw in Microsoft's source code for nearly any purpose.

309. The non-settling States' technical expert, Professor Appel, testified that Section 4.c would leave Microsoft with some control over what portions of Windows source code any developer could demand to see. (Trial Transcript, April 10, 2002 at 3189-3190.) But Section 4.c states simply that Microsoft must permit reasonable access to "the source code and any related documentation and testing suites of Microsoft Platform Software." It does not state that Microsoft may provide reasonable access to only *portions* of such source code.

310. The upshot of Section 4 is that it would provide Microsoft's platform competitors with royalty-free rights to a wide range of Microsoft technology, especially Windows technology. One of the chief proponents of such relief, Sun Microsystems, has been trying to obtain rights to Microsoft's technology for years through ongoing legal proceedings instituted in 1998 before the European Union and, more recently, through a federal lawsuit filed in California.

b. Ambiguity and Feasibility

311. Section 4 imposes engineering obligations upon Microsoft—the identification and documentation of a wide variety of interfaces, protocols, routines and so

forth—yet provides Microsoft with insufficient information to perform the task and imposes timing requirements that are not practical.

312. ***Middleware Definitions.*** The NSPR fails to define what software code in Windows (or on non-PC devices) constitutes a "Microsoft Middleware Product." As discussed above, there is no obvious line in Windows defining where the software, such as the "Internet browser," ends and the rest of the operating system begins (in large part because the very same software is used for multiple purposes). Since the NSPR provides no rules for drawing these lines, it provides no guidance concerning *which* interfaces must be disclosed. (*See* Section III.A.2.b above.)

313. The lack of clarity in the definition of "Microsoft Middleware Product" is exacerbated by the fact that the term appears to be defined in a very granular fashion—very small parts of Windows, down to the code for a single subroutine, seemingly could fit the definition. Because the "middleware" definitions refer generically to "software" that offers "services via APIs or Communications Interfaces," (Sections 22.x.ii and 22.w.), Microsoft could be required under Section 4 to disclose interfaces *within* what Microsoft might regard as a component, including interfaces among subroutines.

314. Furthermore, the definition of "Microsoft Middleware Product" could be read to subject progressively deeper parts of Windows to middleware treatment. That is true because Section 4.a.ii would require Microsoft to disclose the interfaces between each "Microsoft Middleware Product" and the rest of the operating system. Once we do that, the software on the "rest of the operating system" side of the interface could be deemed to be a "Microsoft Middleware Product" because it will now expose APIs. In other words, once APIs are exposed, the software deeper in the operating system could satisfy the

Section 22.x.ii(2) definition because it would be "Middleware" that is similar to "Middleware" functionality offered by a competitor to Microsoft, which under Section 22.w would include nearly any functionality offered in any other operating system that exposes APIs.

315. As software in each layer of Windows were deemed to be a "Microsoft Middleware Product," its interfaces to the next deeper level would have to be disclosed under Section 4.a.ii, potentially creating a cascading effect of ever greater parts of Windows being deemed to be "Microsoft Middleware Products." I am not sure what the non-settling States intended in this regard with their definition of "Microsoft Middleware Product," but that is the apparent consequence of the definitions in the NSPR.

316. ***"Timely Manner."*** The requirements concerning *when* Microsoft would be required to provide the information required by Section 4 also raise significant concerns.

317. Under the NSPR's definition of "Timely Manner" (Section 22.pp), Microsoft would be required to publish the relevant information at the earliest of four specified times. One of those times is whenever the information is "disclosed to Microsoft's applications developers." In fact, new platform technologies in Windows often originate from Microsoft's applications divisions. A group of applications developers may create some useful functionality and realize that the entire developer community could benefit from it, making it useful to include the functionality in Windows.

318. Under Section 4, however, a meeting in which Microsoft's platform developers accepted an application group's suggestion to include new functionality in Windows would appear to constitute an immediate violation of the NSPR. The new

platform functionality would be "disclosed" to the applications group, but the developer community at large would not know about it that day, or anytime soon thereafter, and Microsoft's applications developers would have known about it for years. Microsoft typically would need months or even years to refine the new platform technology, review it with outside developers, test it, and prepare documentation. Instantaneous publication of the technology and how it works is not a real world possibility.

319. Section 22.pp also would require that Microsoft publish the information encompassed within Section 4 whenever such information is "disclosed to any third party." That provision would make it very hard, if not impossible, for Microsoft to work with interested third parties early in the development of new APIs. As it stands today, Microsoft values highly the input of outside developers that work in areas where Microsoft is developing new functionality. At the early stages of a development project (before any code is even written), we typically seek feedback from a relatively small number of outside developers. Technology that is under development and very likely to change should not be publicly posted to Web sites (as Section 4 would require). If any disclosure to a third party triggered a general disclosure obligation to the industry at large, we would probably have to stop seeking input from third parties, which would be a real setback to the development process and to developers that build on Microsoft's platform.

320. Similar problems are presented by other aspects of the definition of "Timely Manner." Like any platform vendor, Microsoft needs time to design, develop, test and document new APIs and related technical information. Given the complexity of the technology and changing ISV needs even as the product is under development, there is no fixed time (until a beta version is released) when we know for certain whether various

functions will be in the product once it is finished. The development process entails conceiving of new features, discussing them within and outside of Microsoft, writing some code, discussing that code, testing it, incorporating feedback, perhaps scrapping the code and starting over with a better idea, and so forth. While that work is underway, it would be neither feasible nor beneficial to disclose all the detailed technical information required by Section 4.

c. Consumer Harm

321. Section 4 would harm Microsoft, the computer industry and consumers in at least five ways.

322. First, Section 4 would contribute to the fragmentation of the Windows platform, with all the harm that entails. Section 4 would grant Microsoft's platform rivals free rein to take Microsoft's Windows technology and implement it any way they like on versions of Windows created under Sections 1 and 2.c, on clones of Windows and on non-Microsoft platforms such as Linux. Section 15.b states explicitly that Microsoft may not seek to maintain compatibility in connection with the licenses of Microsoft's platform technology required by the NSPR.

323. Second, Section 4 would reduce the quality and utility of Windows and thus of PCs and applications built on Windows. Section 4 would require Microsoft to disclose and provide detailed information concerning tens (and perhaps hundreds) of thousands of internal interfaces in the Windows operating system that are not designed for external use. These interfaces are not likely to work well when called by software outside of Windows. For example, internal interfaces typically have no input checking or error handling capability and thus will likely return errors when fed incorrect information. As

noted above, Section 4 also would make it difficult for Microsoft to obtain valuable developer input into Microsoft's development of new platform technology.

324. Third, Section 4 would make it hard for Microsoft to develop new versions of Windows—especially when read in conjunction with Section 5. Creating a new version of Windows to improve performance and fix bugs requires writing a lot of new code, which eliminates many internal interfaces and changes others. Under Section 4, however, such interfaces would have been disclosed, and third party software developers may have relied on them. If Microsoft changes the interfaces, software programs that rely on them will no longer operate properly, which would make Windows less appealing as a platform *and* trigger potential violations of Section 5.

325. Microsoft goes to great lengths today to maintain backward compatibility with existing versions of Windows when it releases a new version of the operating system. We can do that—at great cost—because we have a well-defined, understood set of APIs that we can focus on maintaining. Under Section 4, however, we would be faced with the Hobson's choice of either breaking applications that rely on internal interfaces or trying not to change any disclosed internal interface. Either path would make it more difficult to develop new versions of Windows while maintaining product quality.

326. Fourth, Section 4 would make it hard for Microsoft to satisfy the market's demand for more secure versions of Windows for the reasons set forth above and another important reason: nothing in the NSPR would allow Microsoft to withhold information necessary to ensure the security of Windows, such as cryptographic keys. If you were required to give everyone a key to your house, your home would not be very

secure, and the same is true of Windows. Security concerns will be addressed more fully in the testimony of other Microsoft witnesses.

327. Fifth, Section 4 would greatly reduce Microsoft's incentives to invest in the development of new innovations in Windows—innovations that are essential if Microsoft is to help take computing to the next level. Section 4 largely strips Microsoft of the opportunity to earn any economic return on its investment in innovation. Microsoft must provide its innovations to its competitors even before they are commercialized in Microsoft platform software, and Microsoft's platform software itself would be impaired in the ways I discuss above.

328. An example of the disincentive to innovate created by Section 4 is Microsoft's investment in XML Web Services. As discussed above, Microsoft is attempting to develop the leading platform for building XML Web Services. Thanks to work Microsoft began in the mid-1990s, long before "XML" became a common term in computer circles, we are ahead of competitors such as Sun and AOL. Yet it would not be economically sensible for Microsoft to continue investing in the development of an XML Web Services platform if Microsoft were required to provide its competitors with all of its intellectual property, including source code, relating to that platform as required by Section 4.

5. Section 5

329. Section 5 would subject Microsoft to potential contempt liability for nearly any change to Windows or other "Microsoft Platform Software." That is because nearly any change to Windows could adversely affect some non-Microsoft Middleware running on Windows—especially if the non-Microsoft Middleware made calls to internal

interfaces disclosed under Section 4. Under Section 5, Microsoft would be prohibited from making any change to Windows that "directly or indirectly" interferes with or degrades the performance of non-Microsoft Middleware unless the change was made "for good cause"—an undefined concept that would allow competitors to second guess nearly every Windows design decision.

330. Section 5 is troubling to me because Microsoft's central mission as a platform developer is to promote the development of a wide range of high quality, compatible software and hardware programs. In fact, Microsoft's business centers around enabling software developers to create great applications, not "interfering" with "performance or compatibility" of those applications.

331. My focus here, however, is upon practical considerations: Section 5 would impose an unrealistic burden on Microsoft, especially if Microsoft is obligated to enable third parties to create their own versions of Windows and to publish information to allow ISVs to call upon internal operating system interfaces that are unsuited for that purpose.

332. When Microsoft releases a new version of Windows, some third party applications (including non-Microsoft Middleware) inevitably will be affected by the changes. An improvement to the operating system will often cause a poorly written application to have problems functioning. In fact, the more significant the improvements to the operating system, the higher the likelihood that applications written to the old way of doing things will be affected in some way.

333. Microsoft devotes huge resources to minimizing compatibility problems that may arise from new operating system releases, such as maintaining old APIs

to the greatest extent possible and enabling a new operating system like Windows XP to run in "compatibility mode." In compatibility mode, Windows XP emulates older operating systems built on the Windows 9x code base, enabling many applications written for those older operating systems to run well. Even so, applications written for older operating systems will not always run properly on new operating systems.

334. The central problem presented by Section 5 is that Microsoft simply has no way of knowing all the ways in which all the thousands of products that might be deemed non-Microsoft Middleware might call into Windows and be affected by changes in Windows. ISVs do not have to deal with Microsoft to create software that runs on Windows. We do not and could not track all the software that is written to run Windows. Yet it is easy to imagine being told that we "reasonably" should have known that a particular change to Windows would affect one program or another.

335. Even if we could somehow determine which non-Microsoft Middleware programs would be negatively affected by changes to Windows, disputes would inevitably arise as to what constitutes "good cause"—without which we are not allowed to make the change. For example, the mere addition of new software code to Windows will make a non-Microsoft Middleware program run slightly slower on a memory-constrained PC. Yet the new software might provide new platform capabilities that are extremely useful to millions of software developers. These are the kinds of engineering trade-offs we must make when designing new operating system products. Under Section 5, such routine engineering trade-offs could subject Microsoft to potential contempt liability.

336. Given the complexity of modern software development, disputes concerning Microsoft's compliance could arise even where any performance or compatibility degradation was the fault of the ISV, not Microsoft. For example, plaintiffs put on testimony during the liability phase from a senior executive at Apple claiming that Microsoft had "sabotaged" Apple's QuickTime software. (Trial Transcript, November 4, 1998 at 57-58; Direct Testimony of Avadis Tevanian at ¶ 97.) We traced the problem to a programming error made by Apple. (Direct Testimony of Eric Engstrom at ¶¶ 8, 91-92.) Similarly, in testimony before the Senate Judiciary Committee in 1998, RealNetworks' Chief Executive Officer testified that Microsoft had harmed RealNetworks' software. Here again, we were able to trace the problem to a programming error by RealNetworks. In both cases, however, Microsoft was forced to bear the brunt of false accusations that it had done something to harm non-Microsoft software. Section 5 would enable competitors to level similar charges against Microsoft in the future.

6. Section 6

337. Section 6, entitled "Ban on Exclusive Dealing," bans much more than "exclusive dealing," sweeping within its ambit routine business contracts that do not entail exclusivity. Section 6 applies to contracts relating not only to Windows, but also to *any* Microsoft product, service, feature or technology—even categories where Microsoft is a new entrant. Section 6 would rewrite contracts already in place today, fundamentally changing settled business expectations, and would transfer Microsoft intellectual property to our competitors. And it would effectively prevent Microsoft from entering into mutually beneficial joint development efforts with third parties. Microsoft would be unwilling to

invest resources in a joint venture with any assurance that the technology developed by the venture would benefit Microsoft rather than its competitors.

a. Section 6.a

338. Section 6.a would prevent Microsoft from entering into or enforcing any existing agreement that "restricts" the other party's "development, distribution, promotion or use" of any "non-Microsoft product, service, feature or technology." A great many routine business contracts necessarily entail some "restriction" on such activities by one party in order to provide the other party (here Microsoft) with a reason to enter into a contract. Put differently, a "restriction" may be just the flipside of one party's commitment to another that provides the basis for a deal.

339. For example, a contract under which Microsoft provides funds to a third party to promote a Microsoft product would ordinarily state just that: advertising purchased with Microsoft's money must promote Microsoft products, not competing products. Yet such an agreement could violate Section 6.a because it would "restrict" the third party from promoting a non-Microsoft product.

340. In some product categories, competition is waged in large part on the basis of obtaining or providing exclusive rights. For example, game console developers such as Sony, Nintendo and now Microsoft (with our new Xbox offering) compete to attract ISVs to build new games exclusively for their consoles (at least for some period of time), just as television networks compete to obtain exclusive rights to programming.

341. Here are just a few examples of beneficial business contracts that apparently would be banned by Section 6.a.:

- Microsoft's online service, MSN, provides co-marketing money to a retailer to promote the MSN service on "end caps" on store

shelves. The retailer is "restricted" from promoting competing online services on end caps—that is the placement for which Microsoft is paying.

- Retailers may promote Microsoft's game console with advertisements stating that a hot new game is available "only on Xbox." Microsoft's agreement with the game ISV "restricts" the ISV from offering its game for a period of time on competing game consoles.
- To improve its home publishing software, Microsoft may obtain rights from a third party to include a collection of "clip art" in the next version of Microsoft's publishing product. The agreement "restricts" the third party from offering the same clip art collection for use in a competing publishing product for a period of time.
- Microsoft and an ISV jointly develop new technology. The joint venture agreement "restricts" the ISV for a period of time from developing competing technology.

All of these examples, of course, could be generalized to other situations.

b. Section 6.b

342. Section 6.b does not relate to any third party's commitment to use, promote or distribute any Microsoft software exclusively or at any other level. Rather, Section 6.b regulates the extent to which Microsoft can limit the uses to which third parties can put Microsoft's copyrighted (and patented) software. Ordinarily, an intellectual property owner such as Microsoft may license its software to a third party to use in one way and not another. Of relevance here, Microsoft may grant a third party the rights to distribute part of Microsoft's platform software with its own product (to provide needed updates to Windows, for example). Such a license might well state that the third party may not use Microsoft's platform software to compete against Microsoft's platform. In other words, Microsoft typically would not grant the licensee the right to use Microsoft's software on non-Microsoft platforms.

343. Section 6.b would strip Microsoft of that basic intellectual property right. Section 6.b states that if Microsoft ever allows a third party to redistribute Microsoft's software, it must allow the software to be used with non-Microsoft platforms. That obviously would create a strong disincentive to allowing third parties to redistribute Microsoft's software, causing harm to ISVs and their customers (and the PC ecosystem generally). Microsoft today licenses various types of software for redistribution by third parties because it is efficient to do so.

c. Section 6.c

344. Section 6.c is also written so broadly that it would ban many beneficial contracts (across all aspects of Microsoft's business), including contracts that promote consumer choice.

345. Section 6.c would ban Microsoft from entering into agreements that provide for a minimum percentage commitment to distribute or promote a Microsoft product even where the contracting party provided a much higher percentage commitment to a Microsoft rival. Consider a case in which a Web site agreed that 100% of the music on its site would be presented to consumers in a non-Microsoft media playback format. The Web site might be willing to agree to make 50% of that music *also* available to consumers in Microsoft's Windows Media format, providing consumers with a choice of two formats. Section 6.c would ban Microsoft from entering into such a beneficial agreement even though it plainly does not exclude any competitor.

346. In other words, Section 6.c does not preserve the benefits of a unique aspect of the digital economy: that an agreement to "distribute, promote or use" one technology in a specified percentage often does not prevent the same firm from distributing,

promoting or using a similar technology at the same time, thereby providing consumers with the benefits of greater choice.

347. Section 6.c also would make it difficult for Microsoft to enter into mutually-beneficial joint development agreements and other forms of joint ventures. Such agreements typically entail some measure of "exclusive" commitment to the technology or product that is the focus of the venture.

d. Section 6.d

348. As I explained in Sections I and II of my testimony, Microsoft's successful business is (and always has been) focused upon developing software that serves as a great platform for building other products. We are very good at it. We *enable* others to develop innovative products; we do not enter into contracts that are meant to "interfere with" or "degrade" the performance of non-Microsoft offerings.

349. I am concerned that Section 6.d could be read to prohibit Microsoft from entering into contracts that encourage ISVs to take advantage of unique innovations in Windows. Such contracts are an important aspect of competition in the platform business: after creating new innovations, a platform vendor needs to encourage ISVs to exploit them so that customers see the benefit (generating more consumer interest in the platform). Such a contract does not restrict a third party from developing any software to take advantage of any features of any other platform software. Nevertheless, Section 6.d could be read to prohibit Microsoft from encouraging use of unique Windows features on a theory that software written to those features will not work as well (at least not without additional engineering work) on platforms that lack such features.

e. Section 6.e

350. Section 6.e would have the effect of *reducing* opportunities for third parties to obtain promotional exposure in Windows for their products or services. Section 6.e states that Microsoft may not enter into any agreement in which an IAP or ICP obtains placement in Windows and agrees to "distribute, promote or use" any Microsoft technology. To make the placement in Windows work technically, however, it is usually the case that the third party would have to use at least some Microsoft technology.

351. For example, Microsoft includes a new feature in Windows that makes it easy to post documents or pictures to a Web site, such as MSN Communities. Any OEM can add Web sites to the list of featured sites when building a PC, and any Web site can add itself to the list later. (Web sites can also use their own software to facilitate the process of "uploading" documents or pictures, and many do so.) This new Windows feature will not work unless the Web site implements the requisite Microsoft technology to enable the Windows feature to upload the user's files.

352. Section 6.e also broadly applies not just to placement in Windows, but also to placement in any "Microsoft Middleware Product." As a result, Section 6.e would make it much harder for IAPs and ICPs to obtain promotional opportunities in a wide range of Microsoft software products, to the detriment of the IAPs and ICPs, Microsoft and consumers.

7. Section 7

353. Section 7 appears to be another version of Sections 1 and 2(c), directed at enabling OEMs and others to deconstruct Windows and thereby fragment the Windows platform. Like those sections, Section 7 appears to proceed upon the false premise that each version of Windows is not really a single, integrated product, but rather

an operating system plus a collection of distinct applications, called "Microsoft Middleware Products." Section 7 would entail all the harms and practical difficulties set forth above with respect to Sections 1 and 2.c.

354. I do not know what Section 7's prohibition concerning providing "an access point" to any "Microsoft Middleware Product" means. As far as I know, that term has no commonly understood meaning in the software industry.

8. Section 8

355. Section 8 could be read to ban Microsoft from competing in any product category. I know such a ban would not be reasonable, and yet that is what the language of Section 8 appears to provide for.

356. It states that Microsoft may not take (or threaten) *any* action that directly *or indirectly* adversely affects anyone based directly *or indirectly*, in whole or *in part*, on any actual *or contemplated* use, distribution, promotion, support, development, etc. of *any* non-Microsoft product, service, feature or technology (not limited to middleware). Under this broad provision, *nearly any act of competition could be seen as an adverse act.* Competing means attempting to maximize sales, which often entails taking sales from a rival (adversely affecting them). At the very least, Microsoft would have no comfort that routine business acts would *not* violate Section 8.

a. Acts Promoting Microsoft Software

357. Under Section 8, Microsoft would be subject to legal risk anytime it entered into any contract relating to the development or promotion of any Microsoft product—even if that contract was silent as to non-Microsoft products. If Microsoft entered into such a relationship, it could be charged with taking "adverse actions" against any

competitors in the relevant category that it did not enter into the same relationship with, and thus did not receive the same "consideration" from, Microsoft. Section 8 explicitly states that Microsoft may not withhold "consideration" from a third party that uses, develops, etc. any non-Microsoft product or service.

358. For example, Section 8 provides that Microsoft may not give or withhold "marketing" and "sales support" based upon whether or not a third party distributes or promotes non-Microsoft products. Needless to say, in the business world a supplier such as Microsoft provides marketing support only to firms that promote its products, not to firms that promote competing products.

359. The same is true of most of the forms of "consideration" listed in Section 8. For example, consider Microsoft's competition with Palm in the development of handheld devices. After initially building its own handheld devices, Palm adopted Microsoft's platform business model, licensing its operating system to a number of OEMs that use it to build compatible devices. Microsoft has also followed that model with its Pocket PC platform. As of today, OEMs such as Sony and Handspring manufacture Palm-based devices. Other OEMs, such as Compaq and Hewlett-Packard, manufacture Pocket PC-based devices.

360. Microsoft provides its Pocket PC licensees with "licensing terms," "technical, marketing and sales support," "product information," "technical information," "information about future product plans," and "permission to display trademarks or logos"—all forms of consideration listed in Section 8. Microsoft does not provide *competitors* to the Pocket PC platform with such "consideration," and Pocket PC OEMs do not receive that kind of consideration from Palm. Microsoft and its OEM customers do not

want to share technical information or information about future product plans with Palm's OEMs because we compete with them to deliver new innovations. Similarly, we certainly would not want Palm's OEMs to use any Pocket PC trademarks or logos on their products. Yet these routine business practices would violate Section 8 because our decision not to provide all this consideration to Palm's OEMs is based (directly or indirectly, in whole or in part) on the fact that they are selling Palm products, not Microsoft products.

361. There are many other scenarios that illustrate how Section 8 could ban Microsoft from engaging in very basic business activities. Consider Microsoft's provision of co-marketing funds to OEMs that ship Microsoft software in a particular category, say, personal finance software. Wanting to allocate its resources to their most efficient use, Microsoft would tend to provide co-marketing funds to OEMs that shipped a high volume of Microsoft's offering in this category, Microsoft Money. Yet if Microsoft provided $50,000 in marketing funds to an OEM that shipped 1,000,000 units of Microsoft Money, a second OEM that shipped only 100,000 units of Microsoft Money—and 900,000 units of Intuit's competing Quicken software—could nonetheless demand $50,000 in co-marketing funds from Microsoft. The second OEM could argue that Microsoft's decision to provide it with less than $50,000 was based in part on the fact that it shipped 490,000 copies of non-Microsoft software.

b. Joint Ventures and other Cooperative Efforts

362. Section 8 is another provision that would make it difficult as a practical matter for Microsoft to enter into joint ventures or other cooperative relationships with third parties in the computer industry. The essence of such arrangements is a combination of the two firms' resources to build something new and sell it, not a sharing of

those resources with everyone else in the industry. Yet if Microsoft provided any "consideration" to a cooperative arrangement, it would violate Section 8 if it did not also provide the same consideration to anyone who competed with the arrangement.

c. Intellectual Property Transfers

363. Section 8 would also significantly impair Microsoft's intellectual property rights, thereby reducing Microsoft's incentive to innovate.

364. By its terms, Section 8 would prohibit Microsoft from limiting use of its trademarks and logos to companies that build products using the Microsoft technology to which those trademarks and logos relate. For example, an OEM shipping PCs with the Linux operating system and Sun's StarOffice, and using no Microsoft software at all, could nonetheless demand the right to display the Windows and Office trademarks on its PCs. (Section 8 would prohibit Microsoft from "adversely affect[ing]" the OEM by "withholding . . . permission to display trademarks or logos.")

365. The OEM might want to display Microsoft trademarks to suggest affinity with popular Microsoft software, such as its shipment of a Linux user interface that mimics the Windows user interface (like the Linux KDE interface does). Consumers would be confused, however, if Microsoft's trademarks were used in connection with non-Microsoft software. Before long, Microsoft's trademarks would be greatly devalued and not useful to consumers.

366. I am also concerned that Section 8 could be read to prohibit Microsoft from instituting an enforcement action against a competitor that infringed a Microsoft patent. Although it may not have been the purpose of Section 8, a patent enforcement action against a competitor could be seen as an "action . . . that adversely

affects" the competitor "based directly or indirectly, in whole or in part" on that firm's decision to produce, use or sell a non-Microsoft product (a product embodying Microsoft's patented invention).

9. Section 9

367. Microsoft has not "retaliated" against any of the firms that have participated in this action. We are continuing to compete and cooperate in various ways with competitors such as AOL Time Warner, Sun, Oracle, IBM, Intuit, Novell and others, just as we always have.

368. I am concerned, however, that competitors could try to use Section 9 as a sword to extract business advantage from Microsoft or as a shield against vigorous competition from Microsoft. Although it would have no basis in fact, any competitor who participated in this action could subsequently argue that any Microsoft "action adversely affecting" them was motivated by their participation in this case.

10. Section 10

369. The heading of Section 10, "Respect for User, OEM and Third-Party Licensee Choices," is a misnomer. Section 10 is another provision that directly impacts software design. As for "choice," Section 10 states that Windows may not even inform users that they have a choice to use built-in Windows features if an OEM or Third-Party Licensee has designated non-Microsoft software to perform the relevant functions. (Section 10 states that Windows may not "prompt the user to change [the OEM's] designation.")

370. One of the challenges in good software design is "discoverability;" that is, figuring out how to design a user interface that will enable users to "discover" new

features without creating endless series of icons, drop down lists and dialog boxes. Section 10, however, would ban Microsoft from designing operating systems that invite customers to use the features of the product.

371. Section 10 would impose severe engineering requirements on Microsoft. Windows is not designed to enable arbitrary software programs from third parties (namely, any "Middleware") to be "plugged in" and take over any function within Windows—*including providing functions to other parts of the operating system*.

372. The engineering difficulty arises in part because the non-settling States have attempted to generalize from the concept of a "default browser" to a new, far broader concept of "Default Middleware." No such concept exists in Windows. The "Default Middleware" concept is not feasible from an engineering perspective and, in any event, is so vaguely described in Section 10 that Microsoft would have no way of knowing how to build software that complied with the provision.

373. In general, when a user wants to use a software program (whether a feature of Windows or a separate application), he or she opens the program by clicking on an icon. There is a second way of opening certain kinds of programs. Microsoft has developed functionality that allows users to designate what software they would like to use generally when working with particular types of files, such as Web browsers with HTML files (which are common on the Web). When the user encounters a specified file type, Windows will generally launch the specified program. To make this functionality work, Windows maintains a little database in the "Windows Registry" that includes assignments of particular programs to particular file types.

374. Section 10 presents three major engineering problems in extending that concept to all "Default Middleware." First, it proceeds on the false assumption that there is some "default" mechanism in Windows that relates to the many (and ever changing) categories of "Middleware" defined in Section 22.w. For instance, there is no way today to set up a "default voice recognition" program or a default "instant messaging system." Neither is associated with particular file types.

375. Second, even as to the categories of software where the "default" concept makes sense, Section 10 does not define *where* in the operating system "default" opportunities must be created. This turns out to be a very tricky problem because the process of making software easier to use often entails eliminating boundaries between products, or providing new capabilities within a single product.

376. Several examples came up in the context of Web browsing during the liability phase. The Windows "Help" system is built on HTML. We do not require users to launch a separate Web browser application just to view Windows Help. Another Windows feature, called "Windows Explorer," enables users to browse information on their hard drive, a local area network or the Internet, all in a single window. That benefit would be lost if Microsoft were required to design the system so that a separate Web browser application would be launched when a user moved from the hard drive to the Internet.

377. A more recent example is provided by media players from both RealNetworks and Microsoft, both of which enable the user to browse the Web (such as a music guide site) and download music from within the player, giving the user an integrated experience. That benefit would be lost for Microsoft's Windows Media Player if we were

required to launch a separate Web browsing application when a user merely wanted to look at the music guide site or download music via HTTP.

378. Absent a clear rule, we would not know how to design Windows to facilitate such "default" status, and third parties would have room to argue that they would like to take over some functions, but not others. There are countless ways in which any particular program might "plug in" to provide default functionality, at varying levels of granularity. For example, the "default browser" in Windows 98 could "take over" the functionality of Windows Explorer, providing a single window view of information on the hard drive and the Internet, but it could not take over functions *within* Windows Explorer. Similarly, a third party can register to be the default media player, in place of Windows Media Player, but one cannot register to provide browsing capabilities *within* Windows Media Player.

379. Third, even if we knew all the places in Windows where we would be required to create "default" options, we would be forced to adopt "least common denominator" functionality wherever those options existed because we would not know the quality or the range of features of the third party software that might be installed to provide services to other parts of Windows. Once again, Windows Help provides a good example. In Windows 98, Windows Help took advantage of enhancements to HTML which are sometimes collectively called "Dynamic HTML." Microsoft had documented all of this functionality, and had worked with various standards committees such as the IETF, W3C and ECMA to standardize these enhancements as well. Netscape was aware of the functionality and the relevant standards, but chose not to implement these features. As a result, the "dynamic" aspects of our Help system would not work properly if we were

required to design Windows to enable Netscape Navigator or any other Web browser to supply HTML support to our Help system. Even more fundamentally, our Help system called the many APIs exposed by our HTML software. Netscape never exposed such APIs for its software, and even if it had, they wouldn't have been the same as Microsoft's software exposed.

11. Section 11

380. Section 11 would prohibit a range of beneficial arrangements that platform developers like Microsoft enter into with developers of compatible products on a routine basis.

381. The process of "evangelizing" new operating system services to developers—a big part of the business of any platform developer—would arguably be restricted by Section 11 because that process often involves attempting to persuade developers to rely on functionality in the platform rather than duplicate that functionality in their own products. In evangelizing its platform, Microsoft provides "consideration" to ISVs in the form of technical information, code samples, and the like. In most cases, an ISV's decision to build on a particular platform will not entail an "agreement" to refrain from competing with the platform, but it could in some circumstances.

382. For example, Microsoft might elect to support a software start-up building some innovative new technology on the Windows platform, taking advantage of interesting new features of the platform Microsoft wishes to showcase. Microsoft might invest in the start-up and provide some combination of business guidance, developer support and promotional support. In return, Microsoft might require that the start-up follow

through on building on, and thereby showcasing, the interesting new feature in Windows, rather than seeking to duplicate it, for a specified period of time.

383. Similarly, Microsoft enters into joint ventures and other collaborative efforts from time to time with developers. If a joint arrangement is intended to develop technology that will complement the Windows platform, we would likely have an agreement that for a reasonable time prohibited the other party to the venture from duplicating the relevant Windows functionality or implementing similar functionality on non-Microsoft platforms.

384. Section 11 could ban many routine "work-for-hire" arrangements. Under a work-for-hire arrangement, Microsoft hires an ISV to build software for Microsoft according to our specifications. It is often necessary to give the ISV access to Microsoft source code and other intellectual property in order to create the new software we need. We have full ownership of the resulting "work-for-hire" code, which we may include in Windows (or any other product) and which may constitute a "Microsoft Middleware Product." Outsourcing such development work, however, often wouldn't make business sense absent an agreement that the ISV employees who worked on our project would not for a reasonable period of time develop for Microsoft's platform competitors software similar to the work-for-hire project or the source code the ISV employees saw while working on the project.

385. Section 11 is of particular concern to me because the key terms—"Middleware"—and "Microsoft Middleware Product"—are defined so broadly. As a result, Section 11 would regulate not only agreements that relate in some way to our desktop

versions of Windows, but also to nearly any Microsoft software that exposed APIs or Communications Interfaces.

12. Section 12

386. Section 12 would obligate Microsoft to give "all" source code for "all" of its "Browser" software away to anyone in the industry who would like it, with full rights to modify and use the technology any way they like, all royalty free. Section 12 would effect a transfer of some of Microsoft's most innovative work, in which we have invested more than $750 million, to the industry at large. In fact, under the NSPR, any development work we did on our "Browser" software would result in a *negative* economic return to Microsoft.

387. The source code for Microsoft's "Browser" technology is *all* of our Web browsing technology. Once the source code is disclosed, there is nothing else of value in our "Browser" development work. Section 12 is akin to a requirement that BMW give the automobile industry free rights to use its engine technology or that Coca-Cola gives the secret formula for Coke to Pepsi (and other soda makers).

388. The requirement that Microsoft turn over its "Browser" technology to its competitors would present many problems, beginning with the non-settling States' failure to state what software constitutes Microsoft's "Browser" software.

a. Ambiguity of "Browser" Definition

389. Section 22.e defines "Browser" to mean (in part) "Internet Explorer 6.0" and its successors. While one might suppose that the term "Internet Explorer 6.0" would refer to a specific set of software files, the question of what software constitutes Internet Explorer has been heavily litigated in this case and never resolved. Therefore, if

Section 12 were entered as relief, Microsoft would not know what software it was obligated to give to the industry, and any decision it made could be subject to challenge.

390. The non-settling States have never taken a clear position on what software constitutes Internet Explorer. Indeed, in the liability phase, the non-settling States appeared to suggest that files that Microsoft would think of as part of Internet Explorer, such as its HTML display software and HTTP support, were instead part of the operating system. (*See* Plaintiffs' Revised Proposed Findings of Fact at ¶¶ 161.2.2, 154.3.2.) Absent some clearly articulated rule, there is no answer to the question of whether particular files in Windows are part of the "operating system" or the "browser" because there is no technical or logical distinction between the two. It is all just software that provides useful functions that can be used in various scenarios, including Web browsing. But the very same files that provide functions useful for Web browsing, such as displaying HTML, *also provide* functions useful in other contexts, such as presenting "Help" information or a "richer" view of files on a PC's hard drive.

391. For the remainder of this section, I will assume a definition of Internet Explorer includes the set of files that Microsoft distributes separately from Windows under the name "Internet Explorer 6.0."

b. Windows Fragmentation and Quality

392. Section 12 also would contribute to the fragmentation of the Windows platform, with all the harm that would entail. (*See* Sections II.A.1 and III.B.1.c.)

393. The Internet Explorer software in Windows provides developers with hundreds of APIs that assist in the creation of Internet-enabled software applications. Section 12 would grant everyone in the industry rights to modify Internet Explorer. If

Internet Explorer were modified in unknown and unpredictable ways, applications written to the Internet Explorer APIs would not function properly. If multiple parties made multiple modifications to Internet Explorer, ISVs and Web site developers would need to spend time and money ensuring that their products work with multiple versions of Internet Explorer.

394. This fragmentation problem is compounded by the fact that OEMs and others could swap Microsoft's original version of Internet Explorer out of Windows in favor of any modified version. Developers thus would have no assurance that the platform capabilities provided by Internet Explorer would be present on a PC running Windows.

395. Wholly apart from platform fragmentation concerns, Microsoft could not control the quality of Windows if everyone in the industry has free reign to modify the Internet Explorer software in Windows. Certainly we could not properly test Windows if dozens of companies were offering different versions of Internet Explorer that might or might not provide the same functions to other parts of Windows as Microsoft's version of Internet Explorer does and might or might not provide those functions in the same way.

c. Reduced Innovation and Competition

396. There would be little reason for Microsoft to continue investing in "Browser" software if Section 12 were in effect because Section 12 would divest Microsoft of any significant opportunity to seek reward from its investment. Any new innovation we developed would go straight to our competitors. Section 12 would not even provide Microsoft with a period of time in which to commercialize its development work before handing it over to the industry at large. Section 12 states that Microsoft must give its technology away no later than six months *before* the technology is released to consumers.

397. Under the NSPR, any development work we did in our "Browser" software would be a money-losing proposition. Section 1 would permit anyone who licenses 10,000 copies of Windows to "remove" Microsoft's "Browser" code, claim a reduction in their Windows royalty, and then restore an identical copy of Microsoft's "Browser" licensed from a Section 12 licensee. (*See* Section III.B.1.d above.)

398. Free rights to Microsoft's "Browser" technology would benefit Sun and AOL—companies that have done relatively little to advance the state of Web browsing software. Sun dabbled with a Web browser project called "Hot Java" in the mid-1990s, but dropped it. AOL bought Netscape in 1998 and then failed to release a major new version of Navigator until December 2000, even though AOL has a ready audience for that software in the form of the 35 million subscribers to its online service. AOL's December 2000 release of Navigator was generally not well received by product reviewers or consumers. In the meantime, Microsoft continued to invest heavily in improving its Web browsing software, coming out with Internet Explorer 5.0 and Internet Explorer 6.0, both of which won many awards. Among other innovations, Internet Explorer includes new features, worked out in conjunction with various State Attorneys General and other government officials, to promote consumers' privacy when using Web sites. No other browser that I am aware of includes such privacy features today.

399. Section 12 would also require Microsoft to provide AOL (and the rest of the industry) with the source code for MSN Explorer 6.0 and its successors. MSN Explorer 6.0 is innovative software that makes it easy and enjoyable to use Microsoft's MSN family of Web sites (links are available via www.msn.com).

400. AOL has developed and broadly promoted its own client software for accessing the flagship AOL online service, distributed on millions of CD-ROMs throughout the country. Access to AOL's proprietary online content is generally available only to AOL subscribers and then only through AOL's special client software. Under Section 12, AOL would be free to use Microsoft's innovations in MSN Explorer in its own access software, with no compensation to Microsoft.

401. Reducing Microsoft's incentive to innovate would reduce competition in Web browsing software. Why would AOL continue development of its own Web browsing software if Microsoft's technology were available free of charge, with rights to all improvements (assuming Microsoft made any) for the next ten years?

13. Section 13

402. Section 13 singles out proprietary technology from a particular company, Sun Microsystems, and grants it special treatment. Section 13 would require Microsoft to include a version of Sun's "Java runtime environment" with all copies of Microsoft's Windows Operating System Products and Browsers for ten years. In addition, the Java runtime environment included with Windows must comply with technical specifications established by Sun, eliminating the possibility that the version of Java included with Windows could provide any competition to Sun.

403. Section 13 appears to serve no practical purpose. OEMs are already free to install any Java runtime environment they want on their new PCs. Microsoft makes its Java runtime environment freely available to OEMs and many install it today. OEMs also can install Java runtime environments provided by Sun, IBM or anyone else. Section 13 appears to add nothing to this situation unless it is intended to *require* OEMs to install

Java. But Section 13 presumably would not obligate to Microsoft to require OEMs to install Java since Sections 1, 2.c and 7 enable OEMs to remove any "Middleware distributed by Microsoft" from Windows. (Section 22.x.ii.)

404. Although Section 13 does not appear to serve any purpose, it would create significant problems and risks for Microsoft because it grants Sun unfettered control to define the software that Microsoft must include in Windows. In particular, Section 13 requires that the Java software in Windows comply with the "latest Sun Microsystems Technology Compatibility Kit." The "Compatibility Kit" is developed by Sun. Microsoft software will not satisfy the "Compatibility Kit" because doing so requires a license from Sun granting the right to develop software compliant with the current versions of the Java specifications, and Microsoft no longer has such a license. Indeed, the only software that is likely always to adhere fully to Sun's specifications in the future is Sun's technology. Thus, as a practical matter, Microsoft's only option under Section 13 would be to ship Sun's version of Java software.

405. Section 13 would present a number of significant problems. First, Section 13 would expose Microsoft to substantial intellectual property infringement risk under patent, copyright, trade secret and trademark law. Microsoft would be subject to this risk for any Java software we are required to include in Windows. In the case of a product like Windows, about 120 million copies are distributed annually around the world, making this a very big risk. If Java software from Sun or IBM is alleged to violate any patent, Microsoft could be subject to damages claims and an injunction to block the release of a new version of Windows, to require modifications to existing versions of Windows and to recall Windows products already distributed—potentially causing great damage to

Microsoft as well as OEMs and ISVs. In fact, Kodak recently filed a lawsuit alleging that Sun's Java technology is violating three Kodak patents. *See Eastman Kodak Co.* v. *Sun Microsystems, Inc.,* W.D.N.Y. No. 6:02 civ. 6074 (Telesca, J.)

406. Second, Section 13 places no limits on how Sun might elect to define the specifications for a "compliant" version of Java. Sun could easily include specifications that would render any compliant Java runtime environment incompatible with Windows. Sun could even draw up specifications that would require any compliant Java runtime environment to make changes to Windows that would affect its performance.

407. Sun and Microsoft sued one another in 1997 for claims related to Microsoft's implementation of Java technologies pursuant to a license agreement from Sun. That litigation was settled in January 2001. In June 2000, while the litigation was still pending, Sun sent Microsoft a version of its Java technology (version 1.1.8) that included tests that specifically detected Microsoft innovations in our Windows Java runtime environment. These tests did nothing to test the functionality of our implementation of Java; they simply detected Microsoft's Java innovations and the test "failed" if the innovations were found. Microsoft promptly rejected those tests as a breach of our license agreement with Sun. Under the NSPR, however, nothing constrains the tests Sun might include in some future Java Compatibility Kit except, perhaps, the so-called Java Community Process that was established by Sun and over which I understand Sun has retained veto power for important decisions.

408. Third, Section 13 provides Microsoft with no way to control the size of the software it would be required to include in Windows and Internet Explorer. Implementations of software written to Sun's specifications as it evolves over ten years

could be very large—and would have to be large if Sun elected to "specify" more functions than it does today as constituting a compliant Java runtime environment. Such software could easily double the size of Internet Explorer and, over ten years, could rival the size of Windows itself.

409. Fourth, Section 13 provides Microsoft with no way to control the quality of the Java runtime environment code it must include in Windows and no indemnification against legal risk associated with the non-Microsoft software we would be required to ship. The non-Microsoft code could include bugs or other performance problems or expose security risks. Microsoft would have no right to fix flaws in software supplied by third parties, yet it could be exposed to damages claims caused by them.

410. Finally, I am concerned about the relationship of Section 13 to all the other rules and engineering obligations that would be imposed upon Microsoft by the NSPR. The Java runtime environment that Section 13 would require Microsoft to distribute would be a "Microsoft Middleware Product" because it is "Middleware distributed by Microsoft . . . ." (Section 22.x.ii(2).) In addition, various parts of the Java runtime environment would presumably constitute "Microsoft Middleware Products" in their own right because other "Middleware" programs provide functionality similar to parts of the Java runtime environment. (*Id.*) I have not seen anything in the NSPR that would make clear that Microsoft is not obligated to comply with provisions such as Sections 1, 2(c), 4, 10 and 15 as to these non-Microsoft "Microsoft Middleware Products," but clearly we would have little or no ability to do so.

14. Section 14

411. Microsoft Office is Microsoft's second most important product, generating revenue for Microsoft of more than $7 billion in our last fiscal year. As such, Office is enormously valuable technology in its own right. In addition, the availability of that technology on Windows is an important reason why consumers want to use Windows.

412. Although Office is the quintessential application—not an operating system for the Intel x86 architecture—the NSPR would impose a severe penalty on our Office business: a mandatory transfer to three competitors of rights to all of Microsoft's Office technology, including all improvements to that technology for the next ten years. Like other provisions of the NSPR, Section 14 would provide no compensation to Microsoft for use of its intellectual property, other than a one-time, upfront payment from the "auction" winners that is likely to be low, given the economics of the software industry.

413. Furthermore, the NSPR would authorize the three auction winners to run Office on the "functional equivalents" of Windows that would be enabled by Section 4. In fact, Section 14 gives the auction winners even more help to create clones of Windows than Section 4 does, requiring Microsoft to turn over "all parts of the source code" for Windows "necessary for the porting." In other words, the NSPR would turn over to Microsoft's competitors rights to the two key assets we have built up over many years: the platform capabilities of Windows and the premier application we have built to run on that platform. Taken together they account for roughly two-thirds of Microsoft's revenues.

414. With those assets in hand, and no ongoing royalty to Microsoft, platforms built with Microsoft's technology would have a big price advantage over Windows and Office. Competitors to Microsoft could price their Windows clone software (which could be an API layer on top of Linux) and their version of Office very low, even

free, providing them with a price advantage of several hundred dollars over Microsoft's offerings of similar technology.

415. Professor Shapiro testified that "Microsoft's Office franchise is not directly impacted by [Section 14]." (Direct Testimony of Professor Carl Shapiro ¶ 113.) With all due respect, I strongly disagree. I believe that Section 14 would have a grave impact on Microsoft's Office business.

a. Why Microsoft Built Office

416. As discussed above, the purpose of a software platform is to provide useful system services that developers can use to create great applications. If great applications are written, consumers will buy the platform to run the applications. As such, generating great applications for a software platform is a key aspect of competition in the platform business. Microsoft works to generate a large pool of quality applications for the Windows platform by evangelizing the benefits of the platform to ISVs *and* by building its own applications, such as Microsoft Office, to run on the platform. Of course, developing a useful and successful software product like Microsoft Office is also a good business in its own right.

417. I believe it is important for any platform vendor also to commit itself to building "first party" applications for its platform. Only by sitting in the shoes of applications developers can a platform vendor really learn how to improve its platform. Even more importantly, development of first-party applications allows a platform vendor to showcase the newest innovations in its platform to the rest of the industry, which generates developer and consumer interest in and further use of those features.

418. For example, Microsoft invested heavily in the 1980s and early 1990s in developing versions of Word and Excel for early versions of Windows, when other ISVs had limited interest in the Windows platform. In the early 1990s, we invested heavily to develop a new, 32-bit version of Microsoft Office that took advantage of the many innovations in Windows 95. And today we are investing in new innovations in Office that will take advantage of and thereby showcase the benefits of the .NET platform and future versions of Windows, now under development. Future versions of Office will include innovative new features such as presentation technology that facilitates "digital" meetings (with participants located anywhere in the world), visualizations and other "live" connections to data, and improvements to Outlook so users can spend less time sorting through and "filing" email.

419. Microsoft has invested heavily in Office for nearly twenty years. After a slow start, we have successfully built a business that is a key asset of the company. Sun offers its own productivity suite, StarOffice, which it makes available on many platforms, often for free. StarOffice mimics Microsoft Office in many key respects.

420. I believe it would undermine rather than promote competition to simply transfer important assets such as Office from the leading platform vendor, Microsoft, to less successful platform competitors such as Sun, who are attempting to mimic its products. That is especially true because, as discussed below, the auction of all our existing Office technology plus all the new Office technology we develop over the next ten years, as required by Section 14, would greatly reduce Microsoft's incentives to innovate in business productivity software, further decreasing competition.

421. Although developing great applications for platform software is a key aspect of competition in the platform business, the mere availability of any single application, even an important one like Office, on any particular platform is not likely to turn that platform into a success. Microsoft Office has been available on the Apple Macintosh platform since 1989, yet Apple's sales have remained relatively small when compared with the sales of Windows-based PCs.

b. Reduced Innovation

422. Section 14 would greatly impair Microsoft's incentive and ability to innovate in business productivity software.

423. There is no reason to assume or believe that the "auction" required by Section 14 would generate a significant economic return to Microsoft and, of course, will generate no ongoing revenue stream even though Microsoft's technology would be used for ten years in every version of Office offered by the auction winners. The auction will be akin to a "fire sale" because Microsoft would be required to grant the rights to Microsoft's technology under Section 14—and to not one, but three bidders. Michael Tiemann, the only fact witness to testify on behalf of this provision, suggested that a return to Microsoft of $1.00 would be fair compensation. (Trial Transcript, March 25, 2002 at 1051-52.)

424. The economic return to Microsoft is likely to be low indeed because Section 14 contemplates that all three bidders would get identical rights to identical technology. The winners would thus compete with each other to offer versions of Office on Linux or other platforms. Given the near-zero marginal cost of distributing a unit of software, competition among three companies with identical or very similar software will

tend to drive the price of the software to zero. Since the rights available at auction are unlikely to lead to substantial new revenue for any winner, the bids will be low.

425. At the same time, the existence of three companies competing against Office *with Microsoft's own Office technology* will tend to drive the price of *Microsoft's* version of Office down, for the reason set forth above. Section 14 nominally seeks to protect Microsoft against that revenue loss by providing that the auction winners would be licensed to use Microsoft's technology on operating systems other than Windows and the Apple Macintosh. That limitation will not protect Microsoft, however, because the auction winners would likely serve Windows users—by far the largest customer base—by running their versions of Office on non-Windows server operating systems, making the display available remotely to PCs running Windows. In this way, customers running Windows would have access to Office even though it would not be running locally on their Windows-based PC. Microsoft already makes Office available via servers for customers who prefer that method.

426. Alternatively, the auction winners could simply offer their version of Office on a clone version of Windows. Given all the free rights to Microsoft technology provided for by the NSPR, this would be a very sound strategy. Under the disclosure requirements of Section 4 and the free intellectual property rights provided by Section 15—including Microsoft's operating system patents—it would be relatively easy to develop a functional equivalent to Windows. Section 14 would make it especially easy to develop a version of Office to run on the Windows clone because it expressly grants the Office vendor access to actual Windows source code (showing our programming innovations).

427. Since the Office vendor would get the benefits of Microsoft's R&D work for free, and with no ongoing royalty to Microsoft, it could make both products available at a very low price, perhaps even zero. In fact, I would expect the Office vendor to offer a low-priced or free package of "Windows Clone + Office," just as office productivity software from Corel or Sun Microsystems is usually bundled with Linux or other UNIX distributions.

428. The availability of a reasonably good, low-priced version of Office running on non-Microsoft operating systems would severely hurt Microsoft's operating system business by putting it at a very big price disadvantage. For all the R&D that Microsoft puts into its operating system technology, the economics of the business are such that we generate revenue of only about $70 per Windows unit. We generate revenue of roughly $150 to $275 for each user of Office. (As is customary in the software industry, royalty rates for Office vary considerably by version, volume licensed, and channel of distribution.) That means that a computer user that wanted to run a version of Office would have to consider if he or she was willing to pay an additional $150 to $275—*as much as three times the price of Windows itself*—in order to do so on Microsoft's version of Windows. Microsoft could not simply reduce the price of Office to match or beat the price of the non-Microsoft Office version because we would generate insufficient revenue to support new R&D on the product.

429. In short, under Section 14, our Office business would be greatly devalued because multiple firms would offer essentially the same technology, and at the same time Microsoft would receive little compensation for the technology it conveyed through the auction. Without the prospect of an economic return, it would make no

business sense for Microsoft to continue investing in the development of Office as we have for so many years. The economic disincentives created by Section 14 extend to Microsoft's operating system business as well because the premier technology that Microsoft makes available to run on its operating systems would be available on competing platforms at a much lower cost.

c. Consumer Harm

430. If Microsoft were to reduce its investment in improving Microsoft Office, consumers would lose the benefit of future innovations in Office—innovations that likely would have contributed to economic productivity. The same, of course, is true as to Windows.

431. Section 14 would likely also entail consumer confusion and disappointment concerning the source and quality of various versions of Office. Section 14 is ambiguous as to whether vendors of non-Microsoft versions of Office would be free to use Microsoft's Office trademarks and logos to sell their products—although Section 15 suggests they would. (The non-Microsoft Office vendors would claim that use of Microsoft's trademarks was necessary to signify "interoperability" with Microsoft Office.) Regardless of whether they actually used Microsoft's trademarks, the non-Microsoft Office vendors would certainly claim affinity with Microsoft Office, yet there is nothing in Section 14 that would require them to maintain any quality standards or even compatibility with Microsoft Office or with each other.

d. Office for Apple Macintosh

432. For nearly twenty years Microsoft has been the leading ISV for the Apple Macintosh platform. Microsoft began developing business productivity software for

the Apple platform in 1983, releasing Word 1.0 for the Mac in 1984 (when the Mac was first released) and Excel 1.0 for the Mac in 1985. We first released a suite of business productivity applications for the Mac in 1989. Since then, Microsoft has continued to invest in improving its business productivity software for the Apple platform. Microsoft Office 98 for the Mac was very popular with Mac users, as was Office 2001 for the Mac. We recently released Microsoft Office v. X for the Mac, which takes advantage of innovative new features of Apple's new Mac OS X operating system.

433. In addition to requiring Microsoft to auction off its Office technology to three bidders, Section 14 would require Microsoft to continue to invest, for ten years, in developing new versions of Office for the Apple's Mac OS, with "features consistent with Microsoft Office for Windows." Section 14 would obligate Microsoft to invest its resources in this way without regard to the economic or technical viability of doing so.

434. For example, if the Apple Macintosh platform were to lose share in the future—a possibility that cannot be ruled out given Apple's "near death" experience in the mid-1990s—it would be economically inefficient for Microsoft to continue to invest in building applications for the platform. Other changes in business circumstances, such as a decision by Apple to focus on customer segments that generate little demand for business productivity software, might also render it economically unviable to continue to build new versions of Office for the Mac. A lot can happen over ten years.

435. Section 14's requirement that Microsoft ensure that its Office software for the Apple platform provides features consistent with our Office software for Windows is unrealistic because the underlying platforms may well diverge considerably in the future. Office for Windows will take advantage of key innovations in new versions of

Windows. If Apple does not offer similar innovations in future versions of its platform (and at about the same time as Microsoft does), it may not be technically or economically feasible to provide consistent Office features across the two platforms.

436. Even today, the feature sets of Office for Windows and Office for the Mac vary to some extent. Each version offers features not found in the other. Indeed, Mac users like the fact that Microsoft tailors our Mac version of Office to the Apple platform. Microsoft Office v. X for Mac includes such Mac-only features as Formatting Palette, Microsoft Word Data Merge Manager, Microsoft Excel List Manager, and Microsoft PowerPoint Movies. The personal information managers are so different we even give them different names, Outlook on Windows and Entourage for the Mac.

437. Section 14 is likely to reduce innovation in Office for the Apple platform for the same reasons it will likely reduce innovation in the Windows version of Office. If Section 14 devalues Microsoft's Office business so that Microsoft invests less heavily in it, there will be less innovation in both Office for Windows and Office for the Mac.

438. In addition, Section 14 would put the Apple Mac platform at a serious competitive disadvantage. That is because versions of Microsoft's Office technology will be available for hundreds of dollars less on non-Apple platforms than on Apple platforms.

439. I should point out that both Office 98 and Office v. X for the Mac are native Mac applications. By that I mean Microsoft wrote software specifically to the Mac API set, a substantial undertaking. The apparent implication in Section 14 that Microsoft's Office software for the Apple Macintosh is merely a "port" of code we had written for the

Windows version of Office is incorrect. We tried that approach in the mid-1990s, "porting" Office 4.0 to the Apple Macintosh platform with unsatisfactory results.

15. Section 15

440. Section 15 would give competitors (and everyone else in the industry) the ability to demand free access to vast quantities of Microsoft intellectual property. Since Microsoft's products consist entirely of intellectual property, it is no exaggeration to say that Section 15 would be disastrous for Microsoft.

441. Section 15 is devastating because of the sweeping breadth of the provisions to which it relates, as well as the breadth of the defined terms. Section 4 alone would trigger mandatory licensing of a tremendous amount of Microsoft intellectual property, given the very broad definitions of terms such as "Interoperate" and "Microsoft Platform Software."

442. Under Section 15, Microsoft would be required to grant intellectual property licenses relating to (among other things):

- the inner workings of both desktop and server versions of Windows (both internal interfaces and functionality exposed via APIs);
- all or portions of Windows source code, both desktop and server versions;
- rights to modify Windows in many ways, including by removing software code (and thus functionality) from the product;
- the source code for Internet Explorer 6.0 and MSN Explorer 6.0 and their successors over the next ten years;
- the source code for Microsoft Office, including all new versions we develop over the next ten years;

- interfaces, functionality exposed via interfaces and perhaps source code ("reference implementations") relating to *any* Microsoft software running on any computing device that provides functionality similar to any "Middleware" offered by a Microsoft competitor.

443. Section 15 is particularly onerous when applied in connection with the intellectual property confiscations of Section 4, 12 and 14. (*See* Demonstrative Exhibit 4.)

444. First, Section 15 provides that Microsoft will receive no royalty for the mandated licenses to its intellectual property, including its patents.

445. Second, Section 15 makes clear that once Microsoft's technology is provided to the industry at large, competitors are free to use it in connection with making their products interoperable with other non-Microsoft software. The licenses required by the NSPR are not limited to information relevant to interoperability with desktop versions of Windows, and they are not limited to using the information for purposes of ensuring interoperability with desktop versions of Windows.

446. Third, Section 15 makes clear that Microsoft can do little to guard against fragmentation of the Windows platform (and other Microsoft platform technology). *Section 15.b explicitly states that when licensing its intellectual property, Microsoft may not seek to ensure compatibility with any Microsoft software*.

447. Fourth, Section 15 makes clear that the NSPR would facilitate the cloning of Windows. Section 15.b.iii would enable Microsoft's competitors to license Microsoft's platform technology—including patents—and use them to implement Microsoft's platform technology on Linux or other non-Microsoft operating systems.

448. Finally, Section 15 would make it very expensive for Microsoft to obtain broad patent cross-licenses from other companies in the computer industry, which would increase intellectual property infringement risk for Microsoft, OEMs who build PCs based on Windows and customers. Today companies with large patent portfolios, such as Microsoft, often grant cross-licenses to one another in order to avoid costly patent infringement litigation. Cross-license agreements benefit the PC ecosystem and consumers by facilitating wider commercialization of new technologies. If large parts of Microsoft's intellectual property portfolio were provided to the industry royalty free, it would become very difficult for Microsoft to interest other companies in entering into broad cross-license agreements with Microsoft, leading to more litigation.

16. Section 16

449. Section 16 would regulate Microsoft's development of software that implements technical standards established by any "Standard-Setting Body." Under its vague definition, Section 16 will create a disincentive for Microsoft to support industry standards. That would tend to inhibit, rather than promote, interoperability and it is counter to our .NET technology direction based upon implementing industry standards.

450. Microsoft has a strong track record both in supporting industry standards in its software and in contributing to the development of industry standards. Microsoft's products provide state-of-the-art support for dozens of important standards, enabling developers to make use of them in their products with little effort. In the area of Internet standards alone, we provide excellent implementations of TCP/IP, HTTP, FTP, HTML, XML, SOAP, UDDI, WSDL, PPP, POP3, SMTP, PPTP, LDAP, TELNET and others. (*See* Appendix A.) Our implementation of these standards in Windows promotes

interoperability between Windows and non-Microsoft software, both platforms and applications.

451. Microsoft is a leading contributor to the development of industry standards. Microsoft has been at the forefront of the development of the very important XML, SOAP, UDDI and WSDL standards, contributing much of its R&D work in this area to the World Wide Web Consortium and working with IBM and others to rally the industry around these standards. As I mentioned at the beginning of my testimony, Microsoft has developed a new programming language, called C#, that is particularly well-suited to the needs of developers writing XML Web Services software. Microsoft submitted C# to ECMA, and the new language has now been adopted as an industry standard that anyone can build an implementation of. By contrast, Sun has talked for years about making its Java programming language an industry standard but has not done so, having withdrawn Java from every standards body to which it was ever submitted. Now Sun is seeking to accomplish through the NSPR what it could not achieve in the marketplace by having its controlled "Java Community Process" organization declared a "standards body."

452. Section 16 would create a number of serious problems for Microsoft's efforts to support industry standards in its products.

453. First, Section 16 would potentially subject any Microsoft product to regulation, and this regulation is triggered by mere speech. If a product implements an industry standard, we of course need to state that fact publicly. In most cases, there is no realistic option for Microsoft to build support for an industry standard into a product, but not "publicly claim[ ]" that the product complies with the standard.

454. Second, Section 16 appears to proceed on a false premise: that a software product's "compliance" with a technical standard can be determined by objectively verifiable criteria. (Section 16 states that Microsoft "shall comply with that Standard.") In fact, compliance with an industry standard is usually a matter of degree. Standards are often highly complex, and many are not fully defined, leaving room for individual variation in how the standard is implemented. Standards may include specifications that are erroneous or prove to be impractical when implemented in real products. Even where the standard itself is reasonably clear, the extent to which any product actually implements the standard will often be a matter of opinion.

455. In short, it is often the case that no one "fully" implements a given technical standard, as Section 16 would require Microsoft to do. The reference to "De Facto Standards" in Section 16 does not solve these problems. Typically no single "De Facto Standard" emerges where those problems are present; rather, different software developers will build different implementations of the standard.

456. Third, Section 16 would require Microsoft to "fully" implement standards *even before they have been finalized and adopted* by a Standard-Setting body. Yet before finalization, standards are in flux, with various proponents of the standard debating the virtues of one approach or another.

457. Fourth, Section 16 would require that Microsoft "fully" implement a "Standard" when (i) it is merely under consideration by a Standard-Setting Body, (ii) once it is adopted, and (iii) as "modified from time to time." Nothing in Section 16 grants Microsoft *time* to develop new implementations to meet changing specifications for a standard and nothing states *which* products must comply with the standard. Nor does

Section 16 state whether Microsoft can continue to license existing products based on then-existing implementations of the standard. Such ambiguity, and the corresponding risk of contempt liability, would disincent Microsoft from implementing beneficial standards in its products.

458. Fifth, Section 16 fails to distinguish between bona fide standard-setting bodies, such as the Internet Engineering Task Force or the World Wide Web Consortium, and ad hoc groups associated with particular companies or industry alliances, such as the Java Community Process (which Section 22.kk explicitly includes in the definition of "Standard-Setting Body"). The Java Community Process is a group organized by Sun to obtain feedback on Sun's proprietary Java technology and promote that technology. Sun retains veto control. It is not a true industry Standard-Setting Body.

459. Sixth, it would be very difficult—and would cause considerable customer dissatisfaction—if Microsoft were to keep modifying its products to stay compliant with standards as they "may be modified from time to time by the Standard-Setting Body." (Section 16.a.) Sometimes a new version of a standard is not embraced by the industry for various reasons, but Microsoft would always be required to adopt any new modifications to a standard without regard to the customer benefit of doing so.

460. Furthermore, this requirement would subject Microsoft to considerable risk that its competitors would seek to influence what modifications are made to standards in ways that disadvantage Microsoft, knowing that Microsoft would be obliged to implement them nonetheless. That risk is particularly high in view of the NSPR's decision to call out a Sun Microsystems-sponsored organization, the Java Community Process, as a standard setting body.

461. The effect of Section 16 would be to inject a great deal of legal risk into Microsoft's efforts to support industry standards. Microsoft's flexibility to react to changing circumstances and to make decisions it believes are in the best interests of its customers would be replaced by its need to comply fully with the often ambiguous dictates of this provision. The likely outcome would be less support for industry standards and reduced innovation in Microsoft's products.

17. Section 20

462. Section 20 would require Microsoft to provide information to the non-settling States and wait sixty days before making *any* investment or acquisition or obtaining any exclusive license where the transaction involves a person in any of six categories. Section 20 is worded very broadly. It would require Microsoft to comply with its terms as to:

- Simple acquisitions of equipment "assets," such as telephones, PCs, printers, and insurance policies in the ordinary course of business;
- Routine purchases of stocks and other financial instruments in connection with day-to-day management of Microsoft's investment portfolio;
- Routine business transactions that may entail a license that is "exclusive" in some way or a "direct or indirect" acquisition or investment by Microsoft, no matter how small the transaction may be; and
- Investments by third parties (*i.e.*, "indirect investments") in which Microsoft holds any interest, which would include any investment by any of the many publicly-traded companies in which Microsoft's cash reserves may be invested on any particular day.

Put simply, Section 20 would require Microsoft to wait 60 days, after filing reports with the non-settling States, before engaging in the daily transaction of business.

18. Section 21.b

463. Ten years is a very long time in the software industry. Ten years ago, Windows was just beginning to become broadly successful, even as many ISVs focused on writing applications for MS-DOS and IBM's OS/2. Ten years before that the PC industry barely existed. Given the constantly accelerating pace of innovation, I expect we will see more changes in the computing landscape in the next ten years than in any prior ten year period. The rapid pace of change in the computer industry makes predictions concerning the future course of technological and business development notoriously uncertain, even looking out a few years, much less ten. I am very concerned that any remedy in this matter that extends more than five years would subject Microsoft to constraints that would hinder Microsoft's ability to make changes to its technology or businesses that are necessary to respond to the development of new technologies or other changing circumstances.

* * *

464. Microsoft has worked hard for more than 25 years to develop software technologies that have provided real benefits to people at work, at home and at school. Microsoft's efforts have been vitally important to the development of the PC industry and have contributed powerfully to economic growth both in the United States and overseas. I believe that any remedy in this case should preserve both the great consumer benefits of a consistent and evolving Windows platform and Microsoft's ability and incentive to develop innovative products that will help make computers more useful and easier to use in the future.

I declare under penalty of perjury that the foregoing is true and correct. Executed this 18th day of April, 2002.

Bill Gates

Bill Gates

Appendix A

Standards-Based Networking Protocol Support in Windows

Protocol Name	Win 3.x	NT 3.x	Win95	NT 4.0	Win98	Win2k	WinXP
	November 1993	May 1995	August 1995	July 1996	May 1998	February 2000	October 2001
ipx/spx	x	x	x	x	x	x	x
Appletalk		x		x		x	x
NetBIOS	x	x	x	x	x	x	x
NETBT		x	x	x	x	x	x
SMB/CIFS	x	x	x	x	x	x	x
DLC		x	x	x	x	x	
TCP/IPv4		x	x	x	x	x	x
TCP/IPv6							x
FTP		x	x	x	x	x	x
TFTP						x	x
TELNET		x	x	x	x	x	x
DNS		x	x	x	x	x	x
DHCP		x	x	x	x	x	x
PXE						x	x
Gopher			x	x	x	x	x
HTTP 1.0, 1.1			x	x	x	x	x
SSL (2,3), TLS				x	x	x	x
HTTP Auth Kerberos						x	x
POP3			x	x	x	x	x
SMTP			x	x	x	x	x
IMAP					x	x	x
RIP v1/v2					x	x	x
OSPF						x	x
PPP multi protocol		x	x	x	x	x	x
RADIUS						x	x
MPPE		x	x	x	x	x	x
MPPC		x	x	x	x	x	x
BAP						x	x
Multilink				x	x	x	x
EAP, EAPTLS						x	x
SLIP		x	x	x	x	x	x
PPTP				x	x	x	x
PPPOA					x	x	x
PPPOE							x
L2TP						x	x
IPSEC						x	x
SNMP		x	x	x	x	x	x
RSVP					x	x	
SBM						x	
DIFFSERV						x	x
LDAP						x	x

lpd/lpr/lprmon				x		x	x
rsh, rcp, rexec, rlogin, rcmd				x		x	x
T.120			x	x	x	x	x
UPNP V1					x		x
DAV						x	x
RTP						x	x
IGMP		x	x	x	x	x	x
MADCAP						x	x
PGM							x
GENA					x		x
Kerberos						x	x
SPNEGO						x	x
SASL-GSSAPI						x	x
802.1X							x
ATM UNI 3.1, UNI 4.0, lane					x	x	x
Serial (RS-232)	x	x	x	x	x	x	x
Parallel (IEEE 1284)	x	x	x	x	x	x	x
IRDA (tinytp, lmp, comm, lap, ftp, tranp, obex)			x		x	x	x
IEEE 1394					x	x	x
1394 TCP/IP, digital video					x	x	x
IPv4 over 1394					x		x
1394 digital video					x	x	x
1394 Storage (SBP2)					x	x	x
USB			x		x	x	x
USB Modems					x	x	x
USB Telephony						x	x
USB Storage					x	x	x
Bluetooth							x
Bluetooth PAN, RFCOMM, HCRP, HID							x