IN THE UNITED STATES DISTRICT COURT
FOR THE DISTRICT OF COLUMBIA

UNITED STATES OF AMERICA, Plaintiff, v. MICROSOFT CORPORATION, Defendant.	Civil Action No. 98-1232 (TPJ)
STATE OF NEW YORK *ex rel.* Attorney General ELIOT SPITZER, *et al.*, Plaintiffs, v. MICROSOFT CORPORATION, Defendant.	Civil Action No. 98-1233 (TPJ)

DECLARATION OF EDWARD W. FELTEN

I. INTRODUCTION

1. I have testified twice previously in this action. My background and qualifications were presented in my written direct testimony.

2. I have been asked by the Department of Justice to analyze the technical aspects of Plaintiffs' proposed remedies.

3. Among the areas I have considered are how to define various terms useful for specifying remedies, how to distinguish among various types of software (such as operating system software, middleware, and applications), the technical implications of the proposed changes to Microsoft's corporate structure, and the nature of, and means of achieving, interoperation between software programs.

4. My analysis has also included technical issues related to bundling of operating system and non-operating system software; technical issues and problems stemming from Microsoft's ability to withhold technical information from other companies; OEMs' ability to configure Windows; the feasibility of porting applications to non-Windows platforms; and the technical effects of such porting.

5. The remainder of this Declaration is structured as follows. First, I discuss the definitions of technical terms offered by Plaintiffs. Then I consider, in turn, the four specific sections of Plaintiffs' proposal that have technical implications or warrant comment from a computer science perspective: Section 1, requiring changes to Microsoft's corporate structure; Section 3.b, requiring Microsoft to provide documentation and other technical information to other companies; Section 3.g, prohibiting Microsoft from binding certain products to operating systems; and Section 3.a.iii, requiring Microsoft to allow OEMs to configure Windows.

II. DEFINITIONS IN PLAINTIFF'S PROPOSED REMEDIES

6. The Court's Findings of Fact and Conclusions of Law discuss Microsoft's behavior with respect to certain types of software (e.g. "operating system software"). Plaintiffs' proposal uses similar technical terminology. A careful analysis of Plaintiffs'

proposal must begin by considering how Plaintiffs define the technical terms they use,[1] the technical appropriateness of those definitions, and how they relate to the Court's Findings of Fact and Conclusions of Law.

A. TYPES OF SOFTWARE

7. Plaintiffs define an "Operating System"(§ 7.s) to be the code that controls and interacts with hardware devices, offering a platform to other software. This closely follows the definition given by the Court in paragraph 2 of the Findings of Fact. This definition is relevant because the Court found that there is a distinct market for Operating Systems for PCs (Findings of Fact at ¶ 18), and that Microsoft has monopoly power in that market (Findings of Fact at ¶ 33).

8. Plaintiffs define "Operating System Product"(§ 7.t) to be a product consisting of an Operating System and any other software shipped with it. Under this definition, Windows 95, Windows 98, Windows 2000, Millennium, Whistler, and Blackcomb are Operating System Products[2]. Consistent with my testimony and the testimony of other witnesses at trial, these products are properly not considered Operating Systems; they are bundles of Operating System and other software, some of which should be considered separately from the Operating System for the purpose of framing remedies.

9. Plaintiffs essentially define "Middleware"(§ 7.o) to be software that serves as an intermediary between other pieces of software[3]. This is consistent with the Court's definition in paragraph 28 of the Findings of Fact. The definition is relevant because of the Court's findings regarding the competitive effect of Middleware. (Findings of Fact

[1] In this Declaration, I capitalize all terms defined in Section 7 of Plaintiffs' proposal.
[2] They are also Windows Operating System Products.
[3] Plaintiffs also define "Middleware Product" (§ 7.p). This is a nontechnical definition.

at, e.g., ¶¶ 29, 32, 56, 60) Indeed, a major section of the Findings of Fact is entitled "The Middleware Threats."

10. The current state of technology creates the possibility of widespread development of new and innovative types of Middleware, which could provide cross-platform and cross-device foundations for software developers to build upon. Whether this possibility will be realized depends on economic factors that are beyond my expertise. Professor Henderson discusses these factors in her Declaration.

11. Plaintiffs define "Platform Software"(§ 7.x) to be software that is either an Operating System, or Middleware, or a combination of the two. Platform Software is software that offers APIs or Communications Interfaces.

B. BINDING AND END-USER ACCESS

12. Plaintiffs define "Binding"(§ 7.d) to mean bundling a product into an Operating System Product without allowing OEMs and users ready means to remove or uninstall the product.   This definition is relevant because the Court found that Microsoft illegally Bound Internet Explorer to Windows 98. (Conclusions of Law at pp. 25-34)

13. Plaintiffs define "End-User Access"(§ 7.j) as the ability of an end user to invoke Middleware, either directly or indirectly. The inclusion of indirect invocation is important because many of the common mechanisms for invoking software involve an indirect link between the user's action and the invocation of the software. For example, when the user selects a program by clicking an entry on the Start menu, the system responds to this click by opening a file and reading from that file the location of another file, which is then invoked. "End-User Access" is used only in Plaintiffs' proposed

restriction on Middleware Product Binding (§ 3.g), and I discuss its significance below in paragraph 89.

C. PERSONAL COMPUTER

14. Plaintiffs define a "Personal Computer" ("PC") (§ 7.v) to be an Intel x86-based computer (or a computer based on a microprocessor that is a successor to or competitor of an x86-based microprocessor) using a keyboard and display and configured so that its primary purpose is to be used by one person at a time. This closely matches the Court's definition in paragraph 3 of the Findings of Fact. (Plaintiffs use the term "PC" where the Court used "Intel-compatible PC", but the definitions of the two terms are very similar.) This definition is relevant because the Court has ruled that there is a distinct market for PC Operating Systems (Findings of Fact at ¶ 18), that Microsoft has monopoly power in that market (Findings of Fact at ¶ 33), and that Microsoft has illegally used that monopoly power. (Conclusions of Law at pp. 7-34)

D. TYPES OF INTERFACES

15. As defined by Plaintiffs, "APIs" (§ 7.b) comprise all of the mechanisms by which Platform Software on a PC interacts with another component (which may be either software or hardware) on that same PC, along with some mechanisms for interaction with software on other devices. An API is like a specialized language in which Platform Software and the other component speak to each other, allowing the Platform Software and the other component to ask each other to do things, report on the status of previously requested actions, and generally interact with each other. "Calling" an API is like making a statement in this specialized language. APIs are an important and relevant

concept because they describe the means of interaction between components of a computer system (for example, between different software products).

16. The definition of APIs covers both calls made by the other component to the Platform Software, and calls that the other component is required to understand in order to make full use of the Platform Software[4].

17. As defined by Plaintiffs, "Communications Interfaces"(§ 7.f) comprise all of the mechanisms by which Microsoft Platform Software on a PC interacts with software on other computers.

18. I will use the term "Interfaces" to refer collectively to APIs and Communications Interfaces. Because APIs include all means of interaction between Microsoft Platform Software and other (hardware or software) components on the *same* computer, and Communications Interfaces include all means of interaction between the Microsoft Platform Software and components on *other* computers, Interfaces include all means of interaction between Microsoft Platform Software on a PC and any other components. It is important that these terms have this scope, because this allows the decree to treat all means of interaction between Microsoft Platform Software and other components in a consistent manner.

[4] The latter case (calls that the other component is required to understand) may require some explanation, because one typically thinks of the component as calling the Platform Software, and not the other way around. There are important cases, though, in which the Platform Software "calls back" to the other component.
For example, consider an API that contains a call that allows the other component to ask the Platform Software to copy the contents of a compact disk onto the computer's hard drive. The Platform Software may want to notify the other component when the copying is finished, or it may want to keep the other component apprised of the status of the copying. The API might include a "progress notification" call that the Platform Software could use to inform the other component of the progress of the copying. The other component would have to understand this call in order to take full advantage of the API. The use of such "callback" or "notification" calls is a common idiom in API design.
Accordingly, Plaintiffs' inclusion of calls in both directions between the Platform Software and the other component is appropriate for the purpose of describing all means of interaction between Platform Software and other components.

E. TECHNICAL INFORMATION

19. Plaintiffs define "Technical Information"(§ 7.bb) to be all information about Interfaces that is needed to enable the development of other (hardware or software) components to interoperate with Microsoft Platform Software. Because Interfaces include all means of interaction with the Microsoft Platform Software, Technical Information includes no more and no less information than is required by other components to interoperate fully with the Microsoft Platform Software. This is appropriate and necessary to effectuate Plaintiffs' proposals relating to disclosure of Technical Information, which I discuss below.

20. Plaintiffs provide a non-exhaustive list of items (reference implementations, communications protocols, etc.) that may be included in Technical Information. Microsoft and other parties commonly provide appropriate items from this list when they wish to disclose Interfaces. To illustrate the role of the items on this list, I will now provide a few examples of how and why items on the list are needed by developers. For brevity, I will not detail all items on Plaintiffs' list. Justifications similar to those I set forth below are possible for all of the items on the list.

21. A "reference implementation" is documentation in the form of functioning source code. Unlike a production implementation, a reference implementation is designed to maximize simplicity and clarity of expression (in the source code) rather than technical efficiency.

22. A "communications protocol" is a set of conventions for communication between programs, typically across a network. The TCP/IP protocol that underlies Internet communication and the HTTP protocol for distributing Web content are two examples.

23. A “file format” describes a particular way of arranging data in a file on the hard drive of a computer. Whenever a program saves data to the hard drive, the data must be saved in a file format. Some file formats are standardized (e.g. the GIF and JPEG formats for storing photographs and other visual images) and others are proprietary (such as Microsoft Word’s format for storing documents).

24. “Data structure definitions and layouts” are like file formats, except that they apply to data stored in RAM memory rather than on a hard drive.

25. “Error codes” are special values used to indicate that an API call failed due to some erroneous or unusual condition. When a particular error code is used, the software receiving the error code needs to know what the error code means, so it can decide how to cope with the error. For example, a CD-player application may receive an error code saying that there is no disk in the CD drive. If the application knows the meaning of this error code, it can ask the user to insert a disk.

26. “Memory allocation and deallocation conventions” describe the assumptions an Interface makes about when and how memory is allocated and deallocated. Programs often allocate regions of memory in which to place their data; a program is typically responsible for deallocating that memory when it is done using it. When a program is cooperating with Platform Software via an Interface, the program and the Platform Software will each make assumptions about which of them is responsible for allocating and deallocating the memory needed to hold the data that they will share. If the two sets of assumptions are inconsistent, incorrect behavior will result. In order to prevent this, the Platform Software must document its assumptions, so the other program knows what assumptions it should make.

27. “Threading and synchronization conventions” describe the assumptions an Interface makes about the timing of multiple calls to it by the same program. For example, an Interface may function correctly only when calls to it occur one at a time. Alternatively, the Interface may allow several calls to be going on at the same time. If a software developer wants to use the Interface, he needs to know which of these conventions the Interface follows; otherwise, the developer might violate the Interface’s assumptions, or might sacrifice performance by unnecessarily requiring calls to “wait in line” for their turn, rather than being made all at once.

28. “Algorithms for data translation and reformatting” describe how an Interface changes how data is formatted. It is often beneficial to translate data into a format that has some desirable technical property. For example, compressing data reduces the amount of memory required to store it, and encrypting data makes it unintelligible to intruders. If an Interface translates or reformats data, software that uses that Interface may need to know how the data is reformatted.

III. TECHNICAL IMPLICATIONS OF THE REORGANIZATION PROVISION

29. Section 1 of Plaintiffs’ proposal requires changes to Microsoft’s corporate structure. I understand that the reorganization provisions are motivated primarily by economic considerations, such as a desire to change parties’ incentives. These economic issues are outside my expertise, so I will not discuss them. However, I do wish to comment briefly on the technical implications of the reorganization provisions.

A. TECHNICAL IMPLICATIONS OF MICROSOFT’S CONTROL OVER INTERFACES

30. Microsoft designs the Interfaces that other hardware and software products use to interoperate with Windows. Microsoft has the ability to change these Interfaces in each

new version of Windows. (Microsoft also has the power to withhold Technical Information about its Interfaces; I discuss that issue below, beginning at paragraph 51.)

31. In general, an Interface may match the needs of some software well and serve other software poorly. The time and effort required to create and maintain a product depends in part on how well that product's needs match up with the Interfaces it uses. As a result, an Interface can be designed in a way that facilitates the development of some products while hindering the development of others.

32. The same holds true with respect to individual features within products. A particular Interface may tend to facilitate the development of certain features while hindering the development of others.

33. For example, consider an Interface that allows programs to access storage devices such as hard disk drives and CD drives. The Interface might provide a way for the drive to tell the other software that no there is no disk in the drive, or it might not provide such a facility. Omitting such a feature would hinder the development of drives with removable disks (such as CDs), because there would be no easy way to notify the user of a common error condition encountered when using such drives. Drives with nonremovable disks, such as hard drives, would not be inconvenienced by the omission of this feature.

34. Microsoft's control over Interfaces gives it the ability to control how well various products are served by those Interfaces, and consequently gives Microsoft the ability to shape the technical landscape to favor some products or technologies over others. Professor Henderson discusses the competitive impact of this fact, and how Section 1 of Plaintiffs' proposed remedy affects it.

B. Technical Benefits of Porting Microsoft Office

35. Professors Henderson, Romer, and Shapiro discuss the likelihood that the Applications Business will port Microsoft Office to run on non-Microsoft PC Operating Systems, and the competitive benefits of this porting. Porting Office would also provide technical benefits; I will now discuss two of these benefits.

1. Office, If Ported, Would Be Cross-Platform Middleware

36. As I and other witnesses discussed in testifying at trial, Microsoft Office serves as Middleware, providing a broad set of Interfaces to application developers. (Felten 6/10/99am at 58:22-59:4; Devlin 2/4/99am at 41:16-42:3) Microsoft frequently promotes this platform aspect of Office (see GX 2214), and many software developers use it as a foundation for their applications (for example, custom workflow and document processing applications).

37. Microsoft Office can be ported to run on Operating System Products other than Windows. Indeed, Microsoft has already ported Office to Apple's MacOS. It would be possible for the Applications Business to port Office to other, PC-based Operating System Products.

38. If ported to run on non-Windows PC-based Operating System Products (in such a way that the ported versions exposed substantially the same Interfaces as the Windows version), Office would serve as cross-platform Middleware, because the ported versions of Office would be able to support the same applications that the Windows version did.

39. Having ported Office to other Operating System Products, the Applications Business could then add support for more Interfaces to Office, thereby making it even more attractive as cross-platform Middleware.

2. OFFICE, IF PORTED, WOULD FACILITATE SHARING OF FILES

40. Users frequently share files with each other. For example, a user might use Microsoft Word to create a document on one PC, and then email that document to another user (on another PC) so the second user can read or comment on the document. This kind of file sharing depends on the existence of common file formats that can be used by the software on both PCs.

41. Microsoft Word, by default, saves files on disk in a format that Microsoft designed. When Microsoft Word later opens a file saved in this proprietary format, information in the file is used to recreate the state of the original document. As a consequence of the fact that a very large percentage of PC users use Microsoft Word as their document processing software, the Word file format is the de facto (and currently proprietary) standard for sharing editable text documents on the Internet.

42. Microsoft has not fully disclosed the Word file format. Several groups have tried to use reverse engineering to determine the Word file format, but they have had limited success. Some aspects of the Word file format remain a mystery outside of Microsoft.

43. Other document processing programs cannot interoperate fully with Word if their authors do not know the Word file format. For example, suppose a user writes a document in Word, saves it into a file, and sends a copy of the file to a second user. If the second user tries to open the file in another program, that program may be unable to interpret some of the information in the file, and consequently the second user may receive an incomplete or otherwise inaccurate view of the file.

44. Because Microsoft has not fully disclosed the Word file format, non-Microsoft document processing programs currently cannot read all files created by the Windows

version of Microsoft Word. If Microsoft, or the Applications Business, ported Word to another Operating System Product, the ported version would support sharing of files with other Word users more effectively than a non-Microsoft product can.

45. At least two other programs in Microsoft Office, the Excel spreadsheet and the PowerPoint presentation graphics application, also use proprietary file formats that have not been disclosed fully. The same considerations apply to these programs as to Microsoft Word – porting them would similarly improve users' ability to share files across platforms, assuming that the Microsoft file formats remain the de facto standards.

C. IMPLICATIONS FOR TECHNICAL EFFICIENCY

46. As a technical matter, separating Microsoft into an Operating System Business and an Applications Business is unlikely to compromise, and in fact may enhance, technical efficiency. As the evidence in this case illustrates, there is no reason why Operating System developers and other developers need to be housed under the same roof in order to produce technically efficient or "integrated" (in the sense of providing benefits to consumers by working particularly well together) solutions.

47. This fact is illustrated by the relationship between Caldera OpenLinux and its Web browser. Mr. Allchin's videotape demonstration characterized Caldera OpenLinux as offering "OS Integrated Browsing" (screen shot captured as GX 1707). As both Mr. Allchin and I testified, this integration was achieved even though the Linux Operating System and the KDE browser (the "integrated" browser in Caldera OpenLinux) were developed by different organizations. (Allchin 2/1/99pm at 73:5-15; Felten 6/10/99am at 26:1-14) This example illustrates how software products developed by different organizations, and combined later by an OEM or end user, can offer the same integration

provided in multiple products developed by one company. This fact will be reinforced by the disclosure of all Technical Information to developers outside the Operating Systems Business. In short, the reorganization of Microsoft into an Operating Systems Business and an Applications Business need not compromise technical efficiency. Indeed, particularly in concert with Plaintiffs' Technical Information disclosure provisions, I expect exactly the opposite.

IV. PROVISIONS REQUIRING MICROSOFT TO PROVIDE INFORMATION ABOUT INTERFACES

48. Section 3.b of Plaintiffs' proposal requires Microsoft to provide certain information about Interfaces to other companies.

49. I understand Section 3.b to address two competitive problems established by the evidence introduced at trial. First, Microsoft has illegally used its monopoly power in Operating Systems as a weapon to disadvantage non-Microsoft products in other markets(Conclusions of Law at pp. 9-21), and may, if unconstrained, continue to do so to prevent, degrade, or manipulate technical standards. Second, Microsoft has illegally used offers to disclose, threats not to disclose, and actual refusal to disclose Technical Information, to coerce other companies. (Findings of Fact at ¶¶ 84, 90-92, 116, 122, 129, 237, 288, 338-340, 401; Conclusions of Law at p. 19)

A. THE IMPORTANCE OF INTEROPERABILITY

50. "Interoperability" refers to the ability of one program or hardware component to work with another. For example, an application program may interoperate with an Operating System by calling APIs provided by the Operating System, or a client (e.g., desktop or laptop computer) Operating System Product may interoperate with a server

(i.e., a computer whose primary purpose is to provide services to other computers) Operating System Product via a Communications Interface, or a client Operating System Product may interoperate with another client Operating System Product via a Communications Interface.

51. Microsoft's Operating System monopoly gives it the ability to favor Microsoft products in other markets, by refusing to disclose some of the Interfaces supported by Windows. Such a refusal would allow Microsoft to prevent some products from interoperating fully with Windows.

52. Permitting all products to interoperate fully with Windows is necessary to ensure that those products realize their full potential in terms of performance and functionality. As the Court found, "Since Microsoft decides which ISVs receive betas and other technical support, and when they will receive it, the ability of an ISV to compete in the marketplace for software running on Windows products is highly dependent on Microsoft's cooperation." (Findings of Fact at ¶ 338) The same holds for OEMs and independent hardware vendors ("IHVs"): if they need special favors from Microsoft to make their products interoperate fully with Windows, Microsoft will have the ability to shape the technical landscape in their markets.

B. INTEROPERABILITY: AN EXAMPLE

53. An example will help to illustrate the interoperability issue. Microsoft's current Windows Operating System Products support a "file and printer sharing" facility that allows the user to access files that are stored on other computers, such as file servers, and to print files on printers connected to other computers. To accomplish this, Windows uses the Server Message Block ("SMB") protocol to communicate with the file server or

print server. Microsoft's client and server Operating System Products implement the SMB protocol, so that they can act as file and print servers for Windows clients.

54. The details of the SMB protocol are publicly known, so competing operating system products can and do interoperate with Windows by acting as file and print servers for consumers using their desktop PCs. For example, a product called Samba allows Linux servers to act as file and print servers for Windows clients.

55. Suppose, hypothetically, that a future version of Windows used a new file and printer sharing protocol rather than SMB, and that Microsoft refused to disclose the details of that new protocol. This action would give Microsoft the power to choose which server Operating System Products could interoperate with Windows by acting as file and print servers for Windows clients.

56. In this scenario, a user who created a document on his client PC and wanted to print that document on a shared printer nearby would be able to do so only if the print server was running an Operating System Product that Microsoft allowed to act as a print server. The user's experience in using his PC would be adversely affected by Microsoft's refusal to disclose the new protocol.

57. A customer who felt compelled to buy client Windows Operating System Products would therefore additionally be compelled, due to his desire for interoperability, to buy his server Operating System Products from Microsoft or another vendor to whom Microsoft chose to disclose the new protocol. Microsoft's refusal to disclose the new file and print sharing protocol would prevent some competing server Operating System Products from interoperating fully with Windows, and thus would put them at a significant disadvantage.

58. The same interoperability issue applies to the other Interfaces that Windows Operating System Products use to provide services to Middleware and application software on the same PC (or elsewhere). Whenever Microsoft can refuse to disclose Technical Information about an Interface that allows products in another market to interoperate with Windows, Microsoft has the ability to prevent competing products from interoperating fully. Netscape Navigator, from which the Court found Microsoft withheld Technical Information in the aftermath of the June 1995 meeting (Findings of Fact at ¶ 90-92), is a good example.

C. PROVIDING INFORMATION ABOUT INTERFACES IS THE BEST WAY TO ENSURE INTEROPERABILITY

59. Providing information about the Interfaces supported by Microsoft Platform Software is the only effective way to allow ISVs to interoperate with Windows. Other approaches, such as plug-in mechanisms, fall far short of the goal of guaranteeing interoperability. The following analogy will help to illustrate why this is so.

60. Suppose, hypothetically, that I move to Paris. I do not speak French, so from my standpoint the French language is a proprietary Interface that Parisians use to exchange information.

61. Suppose further that we wish to guarantee that I can converse with the natives just as well as any native French speaker can; that is, we want to ensure that I can interoperate fully with the people of Paris.

62. Obviously, the only way to achieve this goal is to allow me to learn how to speak French. None of the alternatives comes close to solving the problem in a comprehensive or systematic way. Phrase books and simultaneous translation are slow and awkward,

and teaching everybody in Paris to speak English is impractical. The only viable strategy involves disclosing the structure and meaning of the French language to me, so I can learn how to speak it.

63. The same principle holds with respect to the Interfaces that software products use to interoperate. There is no hope of interoperating with another software product if the details of the language it expects you to speak are unavailable to you.

D. SECTION 3.B EFFECTIVELY ADDRESSES THE INTEROPERABILITY PROBLEM

64. Section 3.b addresses the interoperability problem by requiring Microsoft to disclose Technical Information which both (a) other companies need to allow their products to interoperate fully with Windows, and (b) is used by Microsoft itself in certain ways; and by allowing other companies limited access to Microsoft's source code for the sole purpose of ensuring interoperability.

65. A program's source code[5] is a form of documentation – it is the most precise description possible of the program's behavior.

66. Compared to source code, written documentation is easier to use and typically contains the answers to the majority of questions that developers are likely to ask. Written documentation also offers an overview of a program, a perspective that is not available in the source code. Thus written documentation is very valuable, even when source code is available.

67. However, written documentation is necessarily imprecise. Even if the documenters are doing their best to be helpful, human languages are inexact tools for

[5] "Source code" is the form in which software is typically read, written, and manipulated by software developers. Source code is translated, by a program called a compiler, into the "binary code" form in which the computer can execute it.

describing software, so no written description of a program can duplicate the information in the source code.

68. Source code is also useful in understanding the behavior of software that contains bugs. Written documentation usually describes the *intended* behavior of the software, but the source code encodes the *real* behavior, bugs and all. Looking at the source code can help a developer understand how to avoid being tripped up by a bug in Platform Software he is using.

69. Moreover, Section 3.b limits the Technical Information that Microsoft must disclose to that information used internally by Microsoft developers. Although this limitation has practical value because it requires Microsoft to disclose only information which it knows has value to its own developers, this limitation makes controlled access to source code all the more important.

70. For these reasons, the most effective way to ensure interoperability with Windows is to provide both written documentation about Windows, and a way to access the Windows source code to handle those cases where the written documentation is insufficient. Plaintiffs' source code access provision, requiring Microsoft to provide a secure facility where other companies can use the source code for the purpose of ensuring interoperability, is an important complement to the Technical Information disclosure provision. At the same time, by ensuring protection of the source code itself, the provisions go no further than is necessary to promote interoperability.

71. The timeliness requirement in Section 3.b is also needed to allow full interoperability. As the Court found,

> Because of the importance of "time-to-market" in the software industry, ISVs developing software to run on Windows products seek to obtain beta releases and

> other technical information relating to Windows as early and as consistently as possible….[T]he ability of an ISV to compete in the marketplace for software running on Windows products is highly dependent on [receiving this information]. (Findings of Fact at ¶ 338)

The timeliness provision requires Microsoft to publish the information as soon as it is used in a Microsoft product or disclosed to any ISV. This prevents Microsoft from handicapping other software developers in the race to market.

E. INTEROPERABILITY, STANDARDS, AND INFORMATION DISCLOSURE

72. Documented technical standards provide many benefits. One technical benefit of such standards is the interoperability they enable.

73. Companies sometimes create proprietary extensions to standards. Proprietary extensions are sometimes helpful and sometimes harmful, depending on the circumstances.

74. Proprietary extensions to a standard are most likely to be helpful when technological development has gotten ahead of the standardization process. If there is consumer demand for a feature, but there is not yet a standard way to provide that feature, a company may implement the feature rather than waiting for the standard to catch up. Ideally, the company will then propose a standard based on its extension, and will participate in the resulting standardization process. This kind of activity is not in opposition to standards, but merely anticipates the standardization process.

75. However, a company might introduce undisclosed proprietary extensions as a competitive stratagem designed to fragment the existing standard. When a dominant firm uses this tactic, it can put existing standards in serious jeopardy. Professor Henderson

discusses how and when Microsoft has an incentive to follow this “embrace and extend” strategy.

76. When a proprietary extension is not disclosed, it cannot contribute to standardization. Undisclosed proprietary extensions to standards generally lead to fragmentation of the standards, because the public standards body, lacking necessary Technical Information, cannot produce a standard consistent with the undisclosed extension. In short, disclosing interfaces promotes standardization. I next provide an example of how Section 3.b, which requires such disclosure, will have this effect.

F. EXPECTED EFFECT ON KERBEROS IN WINDOWS 2000

77. Microsoft’s version of the “Kerberos” protocol in Windows 2000 illustrates the expected effect of Section 3.b.

78. Kerberos is a widely used protocol for improving the security of networked computer systems. Because the Kerberos protocol is a public standard, several platforms implement it, and these platforms can interoperate.

79. Windows 2000 Professional implements a modified version of the Kerberos protocol that includes a proprietary extension to the standard Kerberos protocol. Because Microsoft has refused to disclose its modified Kerberos protocol, non-Microsoft server Operating System Products cannot implement the modified protocol and consequently cannot interoperate fully with Windows 2000 Professional.

80. If Section 3.b is implemented, developers of other server Operating System Products will be able to get the information they need to implement the modified protocol and allow their products to interoperate with Windows 2000 Professional.

81. It has been reported that Microsoft has stated an intention to disclose its proprietary extensions to Kerberos at some point in the future. (RX 22) It is not clear whether or when this will happen, or what conditions Microsoft will impose on companies in exchange for access to the disclosed information. In any case, even if the information is released immediately and unconditionally, Microsoft will have significantly disadvantaged competing server Operating System Products by withholding the information for so long. This illustrates the need for the timeliness requirement in Section 3.b.

82. It is important to note that Section 3.b does not prevent Microsoft from creating and deploying innovative Interfaces. If Section 3.b had been in effect when Microsoft released Windows 2000 Professional, it would not have prevented Microsoft from creating a new version of Kerberos; it merely would have allowed other software to interoperate with Windows 2000 Professional.

G. OVERALL EFFECT OF INFORMATION DISCLOSURE PROVISIONS

83. I conclude that Section 3.b would effectively address the problem of Microsoft withholding Technical Information that other companies need to interoperate with Windows.

V. PROVISIONS ON PRODUCT BINDING

84. The Court found that Microsoft illegally Bound Internet Explorer to Windows (Conclusions of Law at pp. 25-34), that this Binding harmed consumers who did not want Internet Explorer, by causing "performance degradation, increased risks of incompatibilities, and the introduction of bugs" (Findings of Fact at ¶ 173), and that this Binding harmed even those consumers who wanted Internet Explorer, because it

"jeopardized the security and stability of the operating system". (Findings of Fact at ¶ 174)

85. I understand Section 3.g of Plaintiffs' proposal to require Microsoft to undo the illegal product Binding in which it has already engaged, and to refrain from using technical means to Bind Middleware Products to Windows in the future.

A. PRODUCT BINDING IMPOSES TECHNICAL COSTS

86. In general, product Binding imposes the same kinds of technical costs on consumers that the Court found were imposed in the specific case of Binding Internet Explorer to Windows. These technical costs include reduced performance, increased memory use, and increased risk of incompatibility, security flaws, and other types of bugs. This is illustrated by my prototype removal program, which improved performance and significantly reduced memory use by removing End-User Access to Internet Explorer. (Felten 6/10/99pm at 13:1-16:7, 19:7-20:21; Findings of Fact at ¶ 181)

87. Basing technical decisions on a desire to Bind products rather than on technical considerations alone degrades the technical quality of the resulting products. Lowering quality harms all customers.

B. REMOVABILITY REQUIREMENT

88. Section 3.g allows Microsoft to ship a Middleware Product along with Windows, as long as Microsoft makes End-User Access to that product readily removable by OEMs and end users.

89. The Court has found (at least with respect to Internet Explorer) that from a user's point of view, removing End-User Access to a product is the same thing as removing the product:

> The Add/Remove function did not delete all of the files that contain browsing specific code, nor did it remove browsing-specific code that is used by other programs. The Add/Remove function did, however, remove the functionalities that were provided to the user by Internet Explorer, including the means of launching the Web browser. Accordingly, from the user's perspective, uninstalling Internet Explorer in this way was equivalent to removing the Internet Explorer program from Windows 95. (Findings of Fact at ¶ 165)

90. If Section 3.g were implemented, OEMs would have the option of removing a Bound product if their customers did not want it. Customers who preferred to use both products would be able to get them together, but customers who did not want the Bound product would not be forced to take it. All customers would be spared the technical costs imposed by Binding.

91. Section 3.g.ii provides that Microsoft lower the price of Windows for OEMs who remove a Bound product from Windows. The discount is calculated as the ratio of the binary code size of the removed product (as distributed separately) to the binary code size of the original Windows product. Binary code size is an objective and easily measurable quantity that correlates with the cost of developing the products.

C. The Product Binding Provision is not Burdensome

92. To comply with the product Binding provision, Microsoft's future Windows Operating System Products must allow OEMs and end users ready means for removing End-User Access to any Middleware Product. I will use the term "Unbinding" to refer to the development of the means of removing End-User Access to a Bound product.

93. At present, Microsoft is Binding Internet Explorer to Windows 98 and Windows 2000 Professional. Microsoft is Binding several additional products to Windows 2000 Professional, including (at least) Outlook Express, NetMeeting, and Windows Media Player. Each of these products is available in identical (or substantially identical) form separately from any Windows Operating System Product.

94. It would not be difficult for Microsoft to Unbind Internet Explorer from Windows. The Court found that "Microsoft could easily supply a version of Windows 98 that does not provide the ability to browse the Web," and that "it remains possible to remove Web browsing functionality from Windows 98 without adversely affecting non-Web browsing features of Windows 98 or the functionality of applications running on the operating system." (Findings of Fact at ¶ 177) As I discussed in testifying at trial, Microsoft already allows end users to remove about eighty separate components that are shipped with Windows 98. (Felten 6/10/99pm at 6:7-16; GX 1700)

95. The other products Bound to Windows 2000 Professional should all be at least as easy to Unbind as Internet Explorer is. Therefore, it will not be difficult for Microsoft to comply with the product Binding provision.

96. Plaintiffs allow Microsoft six months from the effective date of the decree to bring itself into compliance with the product Binding provision. Given the facts stated above, six months is ample time for Microsoft to make the necessary changes to its products.

D. OVERALL EFFECT OF THE PRODUCT BINDING PROVISION

97. Section 3.g would require Microsoft to undo the illegal product Binding in which it has already engaged, and to refrain from further Binding of Middleware Products to

Operating Systems. This will lead to improvements in the efficiency and reliability of Windows.

VI. PROVISIONS ON OEMS' RIGHT TO CONFIGURE WINDOWS

98. The Court found that Microsoft illegally forced OEMs to refrain from configuring Windows in ways that their customers wanted, (Findings of Fact at ¶¶ 202-209, 225-229; Conclusions of Law at pp. 10-14) and that "These inhibitions soured Microsoft's relations with OEMs and stymied innovation that might have made Windows PC systems more satisfying to users." (Findings of Fact at ¶ 203)

99. I understand Section 3.a.iii of Plaintiffs' proposal is intended to address this problem by requiring Microsoft to allow OEMs to configure Windows in certain ways.

100. OEMs are technically capable of configuring Windows in all of the ways listed in Section 3.a.iii. In fact, OEMs have expressed a desire to make these kinds of modifications, in the apparent belief that doing so would allow them to serve their customers more effectively. (Findings of Fact at ¶¶ 203-205, 210-211, 214)

101. I do not believe that Section 3.a.iii would lead to any degradation of Windows, because OEMs would have no incentive to make Windows worse. As the Court found,

> If an OEM develops a shell that users do not like as much as Windows, and if the OEM causes that shell to load as the default user interface the first time its PCs are turned on, consumer wrath will fall first upon the OEM, and demand for that OEM's PC systems will decline commensurately with the resulting user dissatisfaction. The market for Intel-compatible PCs is, by all accounts, a competitive one. Consequently, any OEM that tries to force an unwanted, low-quality shell on consumers will do so at its own peril.   (Findings of Fact at ¶ 225)

102. Nor do I believe that Section 3.a.iii would lead to any appreciable increase in the fragmentation of the Windows platform. None of the modifications listed in Section 3.a.iii involves disabling or removing any of the APIs supported by Windows.

103. Microsoft itself causes significant fragmentation of the Windows platform. For example, between September 9, 1999 and April 21, 2000, the Windows 98 code on the PC on which I wrote this Declaration was updated at least twenty-nine times. All of these updates were recommended by Microsoft via their Windows Update service. On April 21, 2000, Microsoft was offering and recommending at least eighteen additional updates that I had not yet downloaded and installed. (These numbers do not include the many updates that provide foreign-language support.) If Section 3.a.iii is implemented, Microsoft will continue to be the largest source of Windows fragmentation.

104. I conclude that Section 3.1.3 would give OEMs the freedom to configure Windows, without technically harming consumers.

VII. CONCLUSIONS

105. Overall, I believe that Plaintiffs' proposal, if implemented, would provide technical benefits to consumers.

I declare under penalty of perjury that the foregoing is true and correct. Executed on April 28, 2000, in Washington, D.C.

____

Edward W. Felten