ORIGINAL

FILED U.S. DISTRICT COURT

2005 JUL -5 P 4: 45

DISTRICT OF UTAH

DEPUTY CLERK

Brent O. Hatch (5715)
Mark F. James (5295)
HATCH, JAMES & DODGE
10 West Broadway, Suite 400
Salt Lake City, Utah 84101
Telephone: (801) 363-6363
Facsimile: (801) 363-6666

Robert Silver (admitted pro hac vice)
Edward Normand (admitted pro hac vice)
Sean Eskovitz (admitted pro hac vice)
BOIES, SCHILLER & FLEXNER LLP
333 Main Street
Armonk, New York 10504
Telephone: (914) 749-8200
Facsimile: (914) 749-8300

Stuart H. Singer (admitted pro hac vice)
BOIES, SCHILLER & FLEXNER LLP
401 East Las Olas Boulevard – Suite 1200
Ft. Lauderdale, Florida 33301
Telephone: (954) 356-0011
Facsimile: (954) 356-0022

Stephen N. Zack (admitted pro hac vice)
BOIES, SCHILLER & FLEXNER LLP
Bank of America Tower – Suite 2800
100 Southeast Second Street
Miami, Florida 33131
Telephone: (305) 539-8400
Facsimile: (305) 539-1307

Attorneys for The SCO Group, Inc.

**IN THE UNITED STATES DISTRICT COURT
FOR THE DISTRICT OF UTAH**

THE SCO GROUP, INC. Plaintiff/Counterclaim-Defendant, v. INTERNATIONAL BUSINESS MACHINES CORPORATION, Defendant/Counterclaim-Plaintiff.	**EXHIBIT S8 TO THE DECLARATION OF JEREMY O. EVANS IN SUPPORT OF SCO'S MEMORANDUM IN OPPOSITION TO IBM'S MOTION FOR SUMMARY JUDGMENT ON BREACH OF CONTRACT CLAIMS** **[Docket No. 350]** **(REFILED IN REDACTED FORM)** Case No. 2:03CV0294DAK Honorable Dale A. Kimball Magistrate Judge Brooke C. Wells

Brent O. Hatch (5715)
Mark F. James (5295)
HATCH, JAMES & DODGE
10 West Broadway, Suite 400
Salt Lake City, Utah 84101
Telephone: (801) 363-6363
Facsimile: (801) 363-6666

Robert Silver (admitted pro hac vice)
Edward Normand (admitted pro hac vice)
Sean Eskovitz (admitted pro hac vice)
BOIES, SCHILLER & FLEXNER LLP
333 Main Street
Armonk, New York 10504
Telephone: (914) 749-8200
Facsimile: (914) 749-8300

Stuart H. Singer (admitted pro hac vice)
BOIES, SCHILLER & FLEXNER LLP
401 East Las Olas Boulevard – Suite 1200
Ft. Lauderdale, Florida 33301
Telephone: (954) 356-0011
Facsimile: (954) 356-0022

Stephen N. Zack (admitted pro hac vice)
BOIES, SCHILLER & FLEXNER LLP
Bank of America Tower – Suite 2800
100 Southeast Second Street
Miami, Florida 33131
Telephone: (305) 539-8400
Facsimile: (305) 539-1307

Attorneys for The SCO Group, Inc.

**IN THE UNITED STATES DISTRICT COURT
FOR THE DISTRICT OF UTAH**

THE SCO GROUP, INC. Plaintiff/Counterclaim-Defendant, v. INTERNATIONAL BUSINESS MACHINES CORPORATION, Defendant/Counterclaim-Plaintiff.	**DECLARATION OF MICHAEL DAVIDSON IN SUPPORT OF SCO'S OPPOSITION TO IBM'S MOTION FOR PARTIAL SUMMARY JUDGMENT** **[Docket No. 350]** ***(REFILED IN REDACTED FORM)*** Case No. 2:03CV0294DAK Honorable Dale A. Kimball Magistrate Judge Brooke C. Wells

DECLARATION OF MICHAEL DAVIDSON

1. My name is Michael Davidson and I am employed by the SCO Group, Inc. My office is located at 5616 Scotts Valley Drive, Suite 200, Scotts Valley, CA 95066. Unless otherwise noted, or evident from the context, this declaration is based on my personal knowledge.

2. I graduated from Queens' College, Cambridge, England, in 1977 with a Bachelor of Arts degree after studying Natural Sciences and Philosophy. I subsequently received the degree of Master of Arts in 1981.

3. Upon graduation, I was employed for approximately one year by the Littlewoods Organization in Liverpool, England, as a trainee computer programmer working with Honeywell Level 6 minicomputers running the GECOS Mod 400 operating system. My work involved developing subroutine libraries in support of various projects which were being implemented at the time. These libraries were developed in the Coral 66 programming language and Level 6 assembly language

4. From 1979 to 1983 I was employed by Rank Strand Ltd in London, England as a software engineer working on the development of embedded control systems used for theatre and television lighting. Most of this code was written in assembly language for the Motorola 6800, 6805, and 6809 processors and the DEC PDP-11. From 1982 onwards all of the development work was hosted on a DEC PDP-11 minicomputer running XENIX-11, a derivative of Version 7 UNIX, and from that time onwards I also wrote C language code for a number of different utilities.

5. From 1984 to 1987 I was employed by Micro Focus Ltd in Newbury, England. Micro Focus developed a COBOL compiler and associated programming tools that was available on a large number of different computer systems. I was responsible for porting the Micro Focus COBOL runtime system to many different versions of the UNIX operating system including XENIX, UNIX System III and UNIX System V running on various processors including the Intel 80286 and 80386, Motorola 68000, Zilog Z8000, National Semiconductors 32000, and Western Electric 32000.

6. I began working for SCO in the United States in 1987, and with the exception of a period of approximately 11 months in 1994 have worked for SCO ever since. At SCO I have held positions of, among others, Manager of Development Systems,

Director of System Software and Principal Architect, and my current position is Operating System Architect, reporting to the Vice-President of Engineering for the SCO Engineering Division. I have developed and read hundreds of thousands of lines of operating system code written in C, C++ and assembly languages. I have also spent a significant amount of time reviewing code developed by other developers and providing my review comments.

7. In 1994 I left SCO to work for a company called Visigenic who were developing database visualization software. At Visigenic I had the position of Senior Software Developer and was responsible for implementing various parts of Visigenic's system in the C and C++ programming languages on both Microsoft Windows and various UNIX based operating systems. By the end of 1994 Visigenic decided to abandon its work on data visualization, and I left the company, returning to my old position at SCO in early 1995.

8. In total, I have worked with both the UNIX operating system and the C programming language for over 20 years.

9. I have worked with automated code comparison tools, including the "diff" and "wdiff" programs described below in paragraph 16.

10. I have examined the question of whether the Journaling File System (JFS) initially provided by IBM in various code drops[1] for Linux (hereafter "JFS-Release") and which appears as a standard component in the 2.6.x kernel (hereafter "JFS-Linux") contains any portion of a Journaling File System code found in IBM's AIX operating system – an initial version released in AIX version 4.x (hereafter "JFS-1") and a later version released with AIX version 5.x (hereafter "JFS-2").

11. I have been instructed by counsel that one work is a "derivative work" of another under federal copyright law if it incorporates in some form a portion of the preexisting work and is substantially similar to the preexisting work. In my understanding, and as I use the term in my analysis, a "modification" based on a preexisting work must also incorporate in some form a portion of the preexisting work.

12. When considering whether a portion of code appears in a portion of another code, I used the following three definitions of types of copying:

[1] A code drop is IBM's term for a version or segment of code that was contributed or released

a. Literal copying – where a line in one piece of code exactly matches a line in another piece of code. The line must match word for word and character for character, although differences in "whitespace" (i.e. the number of spaces or tab characters between words) do not affect the consideration of a match.

b. Near-literal copying – where a line in one piece of code nearly exactly matches a line in another piece of code. The line must substantially match word for word and character for character, although differences in "whitespace" do not affect the consideration of a match. Where differences exist, they must be as simple as the renaming of a variable or constructor type. For example:

   i. In JFS-1, a subroutine called "jfs_lockctl" is considered to be substantially similar to match to the JFS-2 subroutine called "j2_lockctl" (because the two are nearly identical in name, use, and function). Similarly, two routines called "iaccess" and "iAccess" in JFS-1 and JFS-2, respectively are considered to be substantially similar.

   ii. In JFS-1, a macro called "VTOIP" is considered to be substantially similar to a JFS-2 macro called "VP2IP" (because the two macros use an identical or nearly identical name, and have the same use and function).

   iii. In JFS-1, a structure definition identified as "struct inode" is considered to be substantially similar to a JFS-2 structure definition called "inode_t" (because the two are identical or nearly identical in name, use, and function, and because there is a type equivalency statement – a "typedef" – which equates the two).

Similar mappings may be made between JFS-2 and JFS-Release, as well as between JFS-Release and JFS-Linux. Literal and near literal code copying can often be detected using automated code comparison tools, as described below.

c. Non-literal copying – where one set of lines in one piece of code embodies the same structure, sequence and organization as in another piece of code. In many instances, a competent programmer can easily evade literal copying restrictions by paraphrasing the text of the restricted source code while copying the non-literal operative elements of that source code. To conclude that non-literal copying exists, one must undertake often time consuming manual code comparisons. As will be seen in the findings described below, there were sufficient literal and near-literal matches to obviate the need for performing much of this structure, sequence, and organization analysis.

13. For copying purposes, code is defined as both the actual C program text as well as included commentary. This is because literal copying of commentary illustrates both "literal copying" as defined in Paragraph 12.a, as well as (at least) "non-literal copying" as defined in Paragraph 12.c.

14. For example, if the commentary in two different code versions accurately describe what the code does, and if a later commentary version is identical to a previous commentary version, the later code (even if different in appearance from the previous code) must perform the same function as the previous code, and is therefore a copy based on the previous code.

15. After reviewing and comparing JFS-1, JFS-2, and JFS-Release, I conclude that the code found in JFS-2 is a derivative of JFS-1, and that the code found in JFS-Release is in turn a derivative of JFS-2. I further conclude that the code found in JFS-Linux is derived from JFS-Release. Therefore, the JFS code present in standard Linux distributions can be traced directly back to the JFS code found in early versions of AIX (*i.e.*, in AIX version 4.x).

16. As explained below, a program called "htmldiff" was used to assist in the comparison of source files by presenting the results of the comparisons in a conveniently formatted side-by-side fashion. The "htmldiff" program uses two common utilities called

"diff" and "wdiff" to find differences in code, by comparing code line-by-line and word-by-word fashion, respectively.

17. The "htmldiff" program was used to compare the following code components:

a. 33,988 lines of code in JFS-1, as found in the source and header files for AIX version 5.1.0, in the directories src/bos/kernel/pfs and src/bos/usr/include/jfs;

b. 57,859 lines of code in JFS-2, as found in the source and header files for AIX version 5.1.0, in the directories src/bos/kernel/j2 and src/bos/kernel/j2/include;

c. 67,559 lines of code in JFS-Release, as found in the source and header files as released by IBM in the "ref" directories of the JFS 0.0.1 code drop for Linux (dated February 2000);

d. Varying numbers of lines of code in assorted code drops, patches, and kernel releases of Linux, culminating in the 32,598 lines of JFS-Linux code provided as a standard file system component of 2.6.x Linux kernels in the fs/jfs source directory.

18. An analysis of JFS-1 and JFS-2 similarities showed that literal or near-literal copies of thousands of lines from JFS-1 were in JFS-2. I therefore believe that JFS-2 is a derivative of JFS-1.

19. Likewise, an analysis of JFS-2 and JFS-Release showed tens of thousands of lines of code were copied in literal or near-literal fashion. I therefore believe that JFS-Release is a derivative of JFS-2 (and therefore, JFS-Release is also derivative of JFS-1).

20. Finally, an analysis of JFS-Release and JFS-Linux showed tens of thousands of lines of code were copied in literal or near literal fashion. I believe that JFS-Linux is a derivative of JFS-Release (and therefore, JFS-Linux is also derivative of JFS-1).

21. A detailed analysis of the code in question was performed using the "htmldiff" code comparison program and by visual inspection of the actual differences and similarities between the files being compared. The "htmldiff" program uses the "diff" and "wdiff" code comparison programs. The "diff" program is available on all

versions of UNIX and Linux, and is used to determine differences in files. The "diff" program compares files on a line-by-line basis. The "diff" program is configurable by runtime switches. For the purposes of this analysis, one of these switches was set so that all whitespace was considered equivalent (because in general, whitespace does not change the operation of C code), and blank space was ignored. However, with the exception of whitespace, lines are considered by the "diff" program to be completely different even if there is a single character difference between the lines. Using the differences in a file as a guide, it is also therefore possible to determine which lines match exactly (because they are the ones that are not marked as being different by the "diff" program). It is this action that the "htmldiff" program performs.

22. The "wdiff" program is an open-source and widely available program for UNIX and Linux that also reports on differences in files. However the "wdiff" program considers differences word-by-word (instead of line-by-line as the "diff" program does), and so is able to resolve similarities and differences in a finer grained way. As with the "diff" program, the "wdiff" program ignores differences in whitespace.

23. The "htmldiff" program uses the output of the "diff" and the "wdiff" programs to produce a colorized, side-by-side comparison of two source code files. The line numbers of each file are shown in the outside border. When two sets of lines are identical, the "htmldiff" program shows those lines with a white background. If there are any differences at all, then the lines are shown with a yellow background (see Figure 1, below).[2]

[2] Yellow appears as a light-gray on a black-and-white printer.

JFS-1 JFS-2

REDACTED

Figure 1

Comparison of JFS-1 and JFS-2

In this example, lines 328-335 in the file on the left side of Figure 1 are identical to lines 214-221 of the file on the right. However, lines 336-337 on the left side of Figure 1 differ in some respects from lines 222-225 on the right side.

24. When differences in the two files are detected, the "htmldiff" program sends the text to the "wdiff" program. A further colorization is done, so that when code is similar (as determined by a file-by-file set of rules used with the programs) the similar code is colored green.[3] However, when no similarities are found, the code is colored red. For example, in line 336 on the left side of Figure 1 and 222 on the right side, the lines are substantially similar, since the only difference is one that is flagged green by the "htmldiff" program.

25. Lines 342-343 on the left side of Figure 1 have no analog on the right side, so the "htmldiff" program colors these lines red, and the lines on the left and right side may be considered truly different.[4]

26. Lines 337 on the left side of Figure 1 and 223-224 on the right side are seen to be substantially similar, but these similarities were not found using the automated code comparison programs. Instead, the code was manually inspected. However, as

[3] Examples of rules can be seen by reviewing paragraph 12.b above.

[4] Both red and green appear as dark gray on a black-and-white printer, so even without color, the differences and similarities between the left and right sides can be seen easily.

described below, most of the JFS code comparison can be completed using the automated code comparison programs, thereby obviating the need for manual code comparison.

27. The "htmldiff" program tool relates a number of statistics about file contents when comparing two source code files. For files that are literal or non-literal copies, the first statistic presents information as to the relative amount of code copied from a prior code version into a subsequent code version. When the source code files differ, a single simple statistic cannot be used, and all the statistics must be considered. The respective meanings of these statistics are:

a. Similarity – this is a rough percentage of how similar two source code files are, using the "wdiff" program to perform the measurement. If "wdiff" reports that two files are 85% or more similar, we assume that one file is substantially copied from the other. For completeness, the similarity is presented as three numbers, such as "46% == avg(21%,71%)".

i. This means that the average similarity of the two files is 46%.

ii. The first file is 21% similar to the second file, and

iii. The second file is 71% similar to the first.

The reason for this unusual pattern may be explained as follows. Assume that two files have 21 lines in common. If the first file is 100 lines long, then the 21 similar lines account for 21% of the total lines of that file. If the second file is only 30 lines long, then the 21 similar lines account for 71% of the total lines. The average of 21% and 71% is 46%. This indicates that some lines match, but it is still necessary to perform a visual inspection of the code in question to determine if the matches are significant.[5] However, if the "wdiff" program reports "83% == avg(82%,84%)", then the files are nearly identical. The next two statistics should then be consulted to determine the importance of the match.

[5] In this context, the term "significant" refers to a qualitative value of a line or word of code. For example, a code line containing only a '}', even though exactly matched, would clearly not in and of itself be "significant".

b. Exact LOC – this is the number of lines of code and commentary that match exactly, using the "diff" program to perform the line-by-line comparison. This statistic does not count blank or empty lines, so this is an exact measure of the volume of code and comments that have been copied from one file to another. However, since some lines of code are similar in *any* C source code, the larger the value of Exact LOC, the greater its importance.

c. Similar LOC – this is an approximation of the number of lines of code and commentary that match partially, using the "wdiff" program to perform the word-by-word comparison. This statistic *does* count blank and empty lines, and computes a fraction of lines based on the number of matching words and the total number of words. This statistic is a good measure of the total volume of code and comments that have been copied from one file to another. However, since some lines of code are similar in *any* C source code, the larger the value of Similar LOC, the greater its importance.

28. In all cases, the numeric values of these statistics reported by the "htmldiff" program are merely an *indication* of the similarity of the code. The actual code must be visually inspected to determine which of the matches are important.

29. Using the "htmldiff" program, the following versions of JFS code were compared:

a. JFS-1 as found in AIX version 4.3.2 compared to JFS-1 as found in AIX version 5.1.0;

b. JFS-1 as found in AIX version 5.1.0 compared to JFS-2 as found in AIX version 5.1.0;

c. JFS-2 as found in AIX version 5.1.0 compared to JFS-Release as found in the initial code drop version 0.0.1 released by IBM to Linux in February 2000;

d. Various versions of JFS-Release as found in code drops to Linux version 0.0.2 through 1.0.14, and Linux commercial release 2.4.21;

e. JFS-Release found in Linux commercial release 2.4.21 to JFS-Linux as found in Linux commercial release 2.6.0.

30. Because different versions of JFS under different operating systems require different files, it is not always possible to perform a complete one-to-one comparison of files. Where noted, files are sometimes added to, deleted from, or renamed within the JFS directory. However, since JFS is a complex subsystem within an overall operating system, such changes are to be expected.

31. The attached exhibits give examples of some of the different kinds of similarities which were encountered between JFS-2 and JFS-1 in AIX version 5.1.0.

32. Exhibit 1 shows a small file in JFS-2 in AIX version 5.1.0 (kernel/pfs/xix_seek.c) that has a large amount of similarity to a file in JFS-1 in the same version of AIX (kernel/j2/j2_seek.c). Of the 56 lines in the JFS-2 file, 36 lines are identical (literal) copies of lines in the JFS-1 file.

33. Conversely, Exhibit 2 shows a small file in JFS-2 in AIX version 5.1.0 (usr/include/jfs/genalloc.h) that has a small amount of similarity to a file in JFS-1 in the same version of AIX (kernel/j2/include/j2_util.h). Analysis of the file shows some similarities, but not enough to be important.

34. Exhibit 3 shows a large file in JFS-2 in AIX version 5.1.0 (kernel/pfs/xix_access.c) that has a large amount of similarity to a file in JFS-1 in the same version of AIX (kernel/j2/j2_access.c). Of the 972 lines in the JFS-2 file, 699 lines are identical (literal) copies of lines in the JFS-1 file.

35. Exhibit 4 shows a large file in JFS-2 in AIX version 5.1.0 (kernel/pfs/xix_init.c) that has a non-obvious similarity to a file in JFS-1 in the same version of AIX (kernel/j2/j2_init.c). The "diff" program is easily confused when the order of lines is changed (or when small differences appear on each line), so although the lines are marked by the "htmldiff" program as being different, lines 61-149 of the JFS-1 file substantially match lines 74-183 of the JFS-2 file. Similarly, lines 40-59 of the JFS-1 file substantially match lines 53-71 of the JFS-2 file. Because of this conservatism on the part of the "htmldiff" program, each comparison must be manually checked for similarities that were missed by the automated code comparison programs. However, the "htmldiff" program *never* reports on the existence of similarities when there are none – it

only fails to report similarities, and thus gives a more conservative count of matching lines of code.

36. Table 1 shows the measure of similarity between JFS-1 as found in AIX version 4.3.2 compared to JFS-1 as found in AIX version 5.1.0. The comparisons examine both the kernel files as well as those found in various utility and helper programs.

37. Where a file exists in one version of AIX but not the other, no comparison is shown. This addition or deletion of files is to be expected between versions of the operating system, and has no special significance in comparing the evolution of JFS.

38. Files such as comsubs.c, delsubs.c, disubs.c, dqsubs.c, isubs.c,, logsubs.c, xix_dir.c, etc., are especially significant. Each of these files is over 85% similar between versions, and each contains thousands of lines of code.

39. Overall, the similarity between the two versions of JFS-1 is very high, with tens of thousands of lines of code literally or near-literally copied from one version to the other. The version of JFS-1 found in AIX version 5.1.0 is clearly derived from the version of JFS-1 found in AIX version 4.3.2

Table 1

REDACTED

Table 1 (cont'd)

REDACTED

Table 1 (cont'd)

REDACTED

Table 1 (cont'd)

REDACTED

Table 1 (cont'd)

REDACTED

40. Table 2 shows the measure of similarity between JFS-1 as found in AIX version 5.1.0 compared to JFS-2 as found in AIX version 5.1.0. The comparisons examine both the kernel files as well as those found in various utility and helper programs.

41. Where a file exists in one version of AIX but not in the other, no comparison is shown. This addition or deletion of files is to be expected between versions of the file system, and has no special significance in comparing the evolution of JFS. Files such as j2_access.c, j2_rdwr.c, j2_seek.c, and j2_util.c are especially significant, with hundreds of identical lines and highly similar structure.

42. While the numerical values of the similarity statistic between JFS-1 and JFS-2 are only moderate, and a visual inspection of the code is required to fully appreciate the similarities between JFS-1 and JFS-2, there is a great deal of similarity between the two versions. In some cases, hundreds of lines of code are copied literally or near-literally, and many structural similarities exist between the two versions (see, for example, Exhibit 3). Although JFS-2 clearly has some major differences in functionality when compared to JFS-1, it is equally evident that much of the code in JFS-2 found in AIX version 5.1.0 is derived from JFS-1 found in AIX version 5.1.0.

Table 2

REDACTED

Table 2 (cont'd)

REDACTED

Table 2 (cont'd)

REDACTED

43. Table 3 shows the measure of similarity between JFS-2 as found in AIX version 5.1.0 compared to JFS-Release as found in the Linux 0.0.1 code drop. The comparisons examine both the kernel files as well as those found in various utility and helper programs.

44. Where a file exists in AIX but not in Linux (or vice versa), no comparison is shown. This addition or deletion of files is to be expected between versions of the file system and operating system, and has no special significance in comparing the evolution of JFS.

45. Kernel files such as jfs_dmap.c, jfs_dtree.c, jfs_imap.c, jfs_logmgr.c, jfs_txnmgr.c, and jfs_xtree.c are especially significant, with thousands of identical lines each.

46. Overall, the numerical values of the similarity statistic between JFS-2 and JFS-Release are very high, with tens of thousands of lines of code literally or near-

literally copied from one version to the other. The JFS-Release found in the 0.0.1 Linux code drop is clearly derived from JFS-2 found in AIX version 5.1.0.

Table 3

REDACTED

Table 3 (cont'd)

REDACTED

Table 3 (cont'd)

REDACTED

Table 3 (cont'd)

REDACTED

Table 3 (cont'd)

REDACTED

Table 3 (cont'd)

REDACTED

Table 3 (cont'd)

REDACTED

47. Table 4 shows the measure of similarity between JFS-Release as found in the Linux 0.0.1 code drop and the first viable version of JFS-Release, version 1.0.0. The comparisons examine both the kernel files as well as those found in various utility and helper programs.

48. Where a file exists in one version of Linux but not in the other, no comparison is shown. This addition or deletion of files is to be expected between versions of the file system and operating system, and has no special significance in comparing the evolution of JFS.

49. Files such as jfs_dmap.c, jfs_dtree.c, jfs_imap.c, jfs_logmgr.c, jfs_txnmgr.c, and jfs_xtree.c are especially significant, with thousands of identical lines each.

50. Overall, the similarity between JFS-Releases is very high, with tens of thousands of lines of code literally or near-literally copied from one version to the other. The JFS-Release found in the 0.0.1 Linux code drop is clearly derived from JFS-2 found in AIX version 5.1.0.

Table 4

REDACTED

Table 4 (cont'd)

REDACTED

Table 4 (cont'd)

REDACTED

51. Although the results of comparing later versions of JFS-Linux are not provided here in tabular form, proceeding forwards from version 1.0.0 of the JFS-Release code to the JFS-Linux version in Linux 2.6.0, *every intermediate version is nearly* identical to its immediate predecessor in the evolutionary progression of JFS. The final version of JFS-Linux is clearly derived from the version 1.0.0 code drop of JFS-Release which, in turn is derived from JFS-2, which is derived from JFS-1.

52. To fully appreciate the way in which the JFS code has evolved it is necessary to compare each successive version of JFS that was created during its development with its prior and subsequent versions, rather than simply comparing the initial and final versions. Such a comparison would require access to all versions of AIX including all intermediate development versions; however, I have been told by counsel that IBM has failed to produce all such versions. Thus, a complete step by step

comparison of JFS from its origin through all early and later versions of AIX (including AIX versions incorporating JFS-1 and JFS-2) to its final form in JFS Release was not possible.

I declare under penalty of perjury that the foregoing is true and correct.

November 22, 2004

[illegible]

Michael Davidson

CERTIFICATE OF SERVICE

Plaintiff/Counterclaim Defendant, The SCO Group, Inc., hereby certifies that a true and correct copy of the foregoing was served on Defendant IBM on the 5th day of July, 2005 by U.S. Mail to:

David Marriott, Esq.
CRAVATH SWAINE & MOORE LLP
Worldwide Plaza
825 Eighth Avenue
New York, NY 10019

Donald Rosenberg, Esq.
1133 Westchester Avenue
White Plains, NY 10604

Todd Shaughnessy, Esq.
SNELL & WILMER LLP
1200 Gateway Tower West
15 West South Temple
Salt Lake City, UT 84101-1004

Laura K. Chaves