UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

DROPBOX, INC. AND MICROSOFT CORPORATION,

Petitioners,

v.

DATANET, LLC,

Patent Owner.

Cases IPR2024-00078 and IPR2024-00079

U.S. Patent Nos. 8,473,478 and 9,218,348

DECLARATION OF ZAYDOON ("JAY") JAWADI
IN SUPPORT OF PATENT OWNER'S PRELIMINARY RESPONSE

IPR2024-00078 and IPR2024-00079
Exhibit 2001
Dropbox, Inc. and Microsoft Corporation v. Datanet, LLC,

TABLE OF CONTENTS

I. INTRODUCTION 1
A. Engagement and Compensation 1
B. Background and Qualifications 2
C. Person of Ordinary Skill in the Art ("POSITA") 7
D. Case Materials Reviewed 8
E. Extrinsic Materials Considered 9
F. Legal Understanding 10
a. Claim Construction 10
b. Anticipation 11
c. Obviousness 12
d. Person of Ordinary Skill in the Art 15
G. Claim Construction 15
H. Declaration Citations 16
I. Right to Supplement 17
J. Summary of Opinions 18

II. OVERVIEW OF THE '478 AND '348 PATENTS 19

III. GROUND 1: CLAIMS 1-6 AND 8-11 OF THE '478 PATENT AND CLAIMS 1-6, 8-20, AND 23-31 OF THE '348 PATENT WOULD NOT HAVE BEEN OBVIOUS OVER KENLEY AND LONG. 21
A. Background 21
a. Data Backup and Management 22
b. Disk Drive Sectors versus Files 25
i. Computer Data Storage Layers 25
ii. Disk Drive Sectors 27
iii. Files 29
c. BIOS 32
B. Long and Kenley Overview 35

a. Long Overview 35

i. Long is based on disk drive sectors, not files. 36

ii. Long handles sectors containing file data or non-file data but does so on sector basis. 38

iii. Long discloses disk drive tracks. 39

iv. Long describes dealing with data storage devices, but not files and not file systems. 39

v. Long's operations and events relate to data storage devices, not files. 41

vi. Long is implemented as part of the BIOS or as a memory resident program. 43

vii. Long teaches detecting access requests to alter data on storage device locations (sectors) but not files. 44

viii. A POSITA would have understood Long as teaching away from file system implementation. 45

ix. In summary, Long is sector based, not file based, and not applicable to files as contemplated in Petitioners' Expert's Declaration.46

b. Kenley Overview 46

C. A POSITA would not have combined Kenley and Long. 47

a. Petitioner's references teach away from the combination of Kenley and Long. 47

i. Schneider-US and Schneider-PCT 47

b. A POSITA would not have had a reasonable expectation of success in combining the references. 50

i. Long's teachings specific to sector-level implementation could not be reasonably expected to succeed in file-level implementation and would have contradicted Long's basic teachings. 50

ii. Long is implemented as part of BIOS or resident program, both sector-based, low-level. 52

iii. Petitioners' Expert's Declaration's proposed Kenley-Long combination would have required substantial redesign and reconstruction of Long. 52

Datanet Exhibit 2001
Dropbox, Inc. and Microsoft Corp. v. Datanet LLC
IPR2024-00078
Page 3 of 140

iv. Schneider-PCT warns that file-level implementation is difficult to implement, prove, and correct, hence teaching no reasonable expectation of success. 53

v. Petitioner's patent application (EX2017) describes the difficulty of using sector-based solutions to retrieve files.....54

vi. A POSITA would have recognized that the proposed modifications would have changed the principle of operation of the modified reference. 56

c. A POSITA would have recognized that the Petitioners' Expert's Declaration's proposed reasons to combine the references were deficient...57

i. Petitioners' Expert's Declaration's Schneider-US citations do not show Long operating at file level. 57

ii. Long can and does operate at the sector level to perform the functions contemplated by Schneider-US within Schneider-US. 61

iii. Schneider-US teaches away from Long and the Move Method as slow. 62

iv. Schneider-US teaches that the Move Method is outside the operating system. 65

v. Petitioners' Expert's Declaration's EX1013, EX1012, EX1010, and Kenley references do not cure Long's deficiencies related to its being sector based or its lack of detection.....66

vi. EX1013 does not cure Long's deficiencies related to its being sector based or its lack of detection.....68

vii. EX1012 does not cure Long's deficiencies related to its being sector based or its lack of detection.....69

viii. EX1010 does not cure Long's deficiencies related to its being sector based or its lack of detection.....70

ix. Kenley does not cure Long's deficiencies related to its being sector based or its lack of detection. 71

x. Petitioner's patent application (EX2016) describes sector-level implementations and sector-level implementations as different and incompatible. 72

xi. Petitioner's alleged motivations to combine are deficient. 74

Datanet Exhibit 2001
Dropbox, Inc. and Microsoft Corp. v. Datanet LLC
IPR2024-00078
Page 4 of 140

IV. GROUND 2: CLAIMS 1-6 AND 8-11 OF THE ’478 PATENT AND CLAIMS 1-6, 8-20, AND 23-31 OF THE ’348 PATENT ARE NOT OBVIOUS OVER CABRERA, SCHNEIDER-PCT, AND WHITING. 81

A. Schneider-PCT 82

a. Schneider-PCT’s Main Feature Used in Petitioners’ Expert’s Declaration.....82

b. Schneider-PCT Deficiency #1: Schneider-PCT teaches away from any file-level implementations. 83

i. Schneider-PCT teaches away from keeping backups of files as they change. 83

ii. Schneider-PCT teaches that file-level implementation is difficult to implement. 84

iii. Schneider-PCT teaches that sector-level implementations and file-level implementations are different and require different approaches, but sector-level implementations are faster. 85

iv. Schneider-PCT teaches that sector-level archival provides better protection than file-level archival.....86

v. Schneider-PCT teaches that sector-level archival is operating system independent. 88

vi. Schneider-PCT teaches away from managing the storage required for backup by file and states that management by time is advantageous.89

vii. Schneider-PCT teaches away from mapping and redirecting writes into the operating system. 90

viii. Summary of Schneider-PCT teach-aways from any file-level implementations 91

c. Schneider-PCT Deficiency #2: Schneider-PCT does not disclose “detecting an instruction by an operating system to perform an operation on an operating file” or “the step of detecting an instruction by a resident program in a computing device for performing an operation on an operating file.” 92

i. Schneider-PCT discloses sector-based write operation interception, not file-based write operation interception. 92

1. First Schneider-PCT Disclosure of Write Operation Interception: EX1020, 11:30-12:8 93

Datanet Exhibit 2001
Dropbox, Inc. and Microsoft Corp. v. Datanet LLC
IPR2024-00078
Page 5 of 140

2. Second Schneider-PCT Disclosure of Write Operation Interception: EX1020, 30:4-11 95

2-1. Schneider-PCT EX1020, 30:4 of 30:4-11 96

2-2. Schneider-PCT EX1020, 30:5-6 of 30:4-11 97

2-3. Schneider-PCT EX1020, 30:6-8 of 30:4-11 97

2-4. Schneider-PCT EX1020, 30:8-9 of 30:4-11 98

2-5. What Is Interrupt 13h (INT 13h)? 98

2-6. Schneider-PCT EX1020, 30:9-11 of 30:4-11 102

2-7. What Is a Disk Controller? 102

3. Third Schneider-PCT Disclosure of Write Operation Interception: EX1020, 23:3-8 106

4. Fourth Schneider-PCT Disclosure of Write Operation Interception: EX1020, 26:7-8 106

5. Fifth Schneider-PCT Disclosure of Write Operation Interception: EX1020, 26:7-8 107

6. In summary, Schneider-PCT discloses sector-based write operation interception, not file-based write operation interception. . 107

ii. Schneider-PCT also references other patents each of which discloses sector-based write operation interception but does not disclose file-based write operation interception. 108

iii. A POSITA would not have relied on Schneider-PCT to modify Cabrera, since Schneider-PCT teaches away from any file-based implementations. 110

d. A POSITA would not have been motivated to modify Cabrera with Schneider-PCT's interception from sector based to file based. 110

e. A POSITA would not have considered and would not have been motivated to use Schneider-PCT to reproduce the '478 and '348 Patents. . 112

B. Cabrera 112

a. Cabrera Deficiency #1: Cabrera does not disclose "detecting an instruction to perform an operation on an operating file." 112

i. Petitioners' Expert's Declaration does not identify "an operation on an operating file," "an instruction to perform an operation on an

Datanet Exhibit 2001
Dropbox, Inc. and Microsoft Corp. v. Datanet LLC
IPR2024-00078
Page 6 of 140

operating file," or "detecting an instruction to perform an operation on an operating file" in Cabrera. 113

ii. Schneider-PCT does not cure Cabrera's deficiency of not disclosing "detecting an instruction to perform an operation on an operating file." 116

iii. Cabrera's archive is performed only after data is already present in the local storage. 117

iv. Cabrera's "any triggering event" is not an "an operation on an operating file" and not "an instruction to perform an operation on an operating file." 119

b. Cabrera Deficiency #2: Cabrera does not disclose "creating an archive file from the operating file and storing the archive file in a temporary [first] storage location temporally proximate to the operation being performed on the operating file and responsive to detecting the instruction." 119

c. Cabrera Deficiency #3: Cabrera does not teach the '478 limitation of "searching the first temporary storage location for the archive file responsive to the occurrence of a first event" or the '348 limitation of "identifying presence of the archive file in the temporary storage location responsive to the occurrence of a first event." 124

C. Whiting 127

a. Whiting's Main Features Used in Petitioners' Expert's Declaration127

b. Whiting Deficiency #1: Other than Whiting's main features used in Petitioners' Expert's Declaration, Whiting is deficient vis-à-vis the claim limitations of '478 and '348 Patents. 127

c. Whiting Deficiency #2: Whiting does not disclose "detecting an instruction to perform an operation on an operating file." 128

d. Whiting Deficiency #3: Whiting does not disclose temporally proximate to the operation being performed on the operating file. 128

D. Motivation NOT to Combine Cabrera, Schneider-PCT, and Whiting...128

a. A POSITA would not have been motivated to combine Cabrera, Schneider-PCT, and Whiting, because Cabrera would need intercepting file write operations at the file level, but Schneider-PCT teaches intercepting write operations at the sector level only. 129

b. A POSITA would not have been motivated to combine Cabrera, Schneider-PCT, and Whiting, because Schneider-PCT teaches away from file-level implementations. 131

V. CONCLUSION.....131

Datanet Exhibit 2001
Dropbox, Inc. and Microsoft Corp. v. Datanet LLC
IPR2024-00078
Page 8 of 140

I, Zaydoon ("Jay") Jawadi, declare as follows:

I. INTRODUCTION

1. My name is Zaydoon ("Jay") Jawadi. I am over eighteen years of age, of sound mind, and qualified to make the statements set forth in this Declaration. With the exception of the legal principles counsel has supplied to me, all the facts and statements contained herein are within my personal knowledge and are in all respects true and correct.

2. I am an independent expert and consultant.

A. Engagement and Compensation

3. I have been retained as an expert witness on behalf of Datanet, LLC ("Datanet") for the above-captioned *Inter Partes* Reviews (IPRs) regarding U.S. Patent No. 8,473,478 ("'478 Patent") and U.S. Patent No. 9,218,348 ("'348 Patent").

4. I am being compensated for my work and any travel expenses in connection with this proceeding at my standard consulting rates. My compensation is in no way dependent on or contingent on the outcome of my analysis or opinions rendered in this proceeding and is in no way dependent on or contingent on the results of these or any other proceedings relating to the above-captioned patents.

B. Background and Qualifications

5. As shown in my curriculum vitae (attached as Exhibit 2002), I have a Master of Science in Computer Science from Columbia University with a Citation for Outstanding Achievement – Dean's Honor Student, a Bachelor of Science in Electrical Engineering from Mosul University, and over 40 years of experience in software and product design and development, engineering, consulting, and management in the fields of data storage, Internet, software, data networking, computing systems, and telecommunication.

6. I have worked with and possess expertise in numerous technologies, including system and embedded software, data storage technologies and interfaces, Internet and website technologies, databases, data networking technologies and protocols, and telephony.

7. From 1978 to 1980, I worked as a telecommunication/electrical engineer for Emirtel (formerly Cable and Wireless, now Etisalat). During my employment at Emirtel, among other things, I worked on telephony and telecommunication products and services, and I designed and developed software in assembly and high-level languages for archiving, storing, and retrieving data to and from data storage devices, such as disk drives and tape drives. I also maintained and operated software programs for critical data archival, backup,

storage, and retrieval using data storage devices, such as disk drives and tape drives.

8. From 1981 to 1983, I worked as a software engineer for Amdahl Corporation (now Fujitsu), a California-based Kenley supplier of computers, systems, and data storage subsystems. Among my responsibilities, I designed and developed bug tracking software using relational databases.

9. From 1984 to 1994, I worked as a software, data storage, and systems consultant to various data storage and computer companies in California, the United States, Asia, and Europe. I provided technical consulting services in database design and development, data storage, data storage systems, data storage devices, software design and development, system software, device driver software, data storage device firmware, disk drive controller firmware, data storage software, data storage chips, data storage tools, data storage test systems and test software, SCSI, IDE, data storage and I/O protocol development systems, data storage controllers, data storage and I/O protocol analyzers, data storage and I/O monitoring systems, and data storage manufacturing systems and software. In one of my assignments, I was the lead developer and team leader of a commercial relational database development project for one of the top 25 companies (then and now) of America's Fortune-500 list.

10. From 1992 to 1996, I was President and founder of Zadian Technologies, Inc., a California-based leading supplier of networked data storage test systems, with over 50,000 units installed worldwide in mission-critical customer operations with premier high-technology customers, such as Conner Peripherals (now Seagate), DEC (now HP), EMC (now Dell EMC), Exabyte, Fujitsu, HP, IBM, Intel, Iomega, Quantum (now Seagate), Seagate, Sony, StorageTek, SyQuest, Tandberg, Tandem (now HP), Toshiba, Unisys, and WD. The company's products included test systems, manufacturing systems, and development systems for data storage devices (disk drives, tape drives, removable drives, flash drives, optical drives, CD-ROM drives, Jukeboxes, and RAID) and data storage interfaces (SCSI, ATA / IDE / ATAPI, Fibre Channel, SSA, and PCMCIA / PC Card).

11. In 1996, Zadian Technologies was acquired by UK-based Xyratex International LTD (NASDAQ: XRTX, which was later acquired by Seagate, NASDAQ: STX, in 2014). Following Zadian's acquisition by Xyratex, I became an employee of Xyratex until 1998. At Xyratex, I was a general manager of a data storage interface business unit and, subsequently, a general manager of a data networking analysis tools business unit, which designed and built Gigabit Ethernet network protocol analysis and monitoring products, which were sold, under OEM

agreement, by the largest supplier of network protocol analysis and monitoring products.

12. From 1999 to 2001, I was CEO, Chairman, and cofounder of Can Do, Inc., a California-based Internet eCommerce and community company. The CanDo.com website offered over 10,000 products for sale as well as extensive consumer features, such as news, chat, messages, and product information for people with disabilities. The company also provided technologies for display magnification and sound/audio adaptation through the Internet to make websites more accessible to persons with vision and hearing impairments. The company was funded by leading venture capital firms.

13. From 2001 to 2007, I was President and cofounder of CoAssure, Inc., a California-based provider of Web-based technology services and solutions for automated telephony speech recognition and touchtone applications, serving multiple Fortune-500 companies. The system was delivered as Software as a Service (SaaS). Among other things, I was responsible for system and file archival, backup, restoration, and management.

14. From 2009 to 2022, I was President and cofounder of Rate Speeches, Inc., a California-based Internet company providing online services, resources, and technologies for creating, rating, evaluating, and enhancing presentation, and communication skills. Rate Speeches also operated the ratespeeches.com website

and the Speech Evaluator online software. The system included functionality delivered as Software as a Service (SaaS).

15. In my career, I have designed, developed, tested, and deployed as well as overseen the design, development, testing, and deployment of multiple complex software, software applications, system software, websites, and systems.

16. Since moving to Silicon Valley in Northern California in 1981, I have worked on numerous technology products that have generated billions of dollars in sales for my employers and customers.

17. I hold a California community college lifelong computer science instructor credential. I have taught various data storage and computer technologies, including SCSI, IDE (EIDE/ATA), SSA, and Fibre Channel, to thousands of professional engineers and academic students in the United States, Europe, and Asia.

18. Additionally, in my work as an expert and consultant, I have examined, analyzed, and inspected numerous data storage systems, computer systems, software products, database software, and websites, spanning tens of millions of lines of software source code.

19. By September 2000, I had more than two decades of expertise directly applicable to the technology at issue in this case, including expertise in file management; **archival, backup, and restoration**; **BIOS** programming; device

drivers; interrupt handlers; interrupt 13h (**INT 13h**) handlers; **trapping / intercepting / detecting** device drivers and BIOS interrupts; resident programs and terminate-and-stay-resident programs; disk drive firmware; **SCSI, IDE**, SSA, and Fibre Channel; **disk drives, tape drives, optical drives, and other storage devices**; **disk drive controllers** and host adapters; database and database application design and development; as well as CP/M, DOS, Windows, Unix, and other operating systems.

20. My additional industry experience is in my curriculum vitae.

21. My expert litigation support cases, including cases in which I have testified during the last four years as an expert, can be found in my curriculum vitae, which is Exhibit 2002.

22. In light of my education, training, knowledge, and experience, I am qualified to provide opinions regarding the state of the art at the time of the invention (or the priority date) of the ’478 and the ’348 Patents, which I understand to be no later than September 21, 2000, and I am qualified to provide opinions regarding how a person of ordinary skill in the art (“POSITA”) at that time would have interpreted and understood the ’478 Patent and the ’348 Patent.

C. Person of Ordinary Skill in the Art (“POSITA”)

23. I have not been asked to opine on the definition of a POSITA for purposes of this declaration supporting the Patent Owner’s preliminary response. I

reserve the right to supplement and offer my opinions on the definition of a POSITA as the proceeding progresses.

24. I understand that Petitioners' Expert has offered a POSITA definition. Exhibit 2003 ¶ 31 (both IPR proceedings). My opinions below do not substantively change if I apply his specific proposed definition.

25. I myself was qualified as at least a POSITA under his (or any other reasonable) definition at the time the '478 Patent was filed and at its priority date as well as at the time the '348 Patent was filed and at its priority date. Over the course of my career, I have also supervised and directed many such persons.

D. Case Materials Reviewed

26. In preparing this Declaration, I reviewed the '478 Patent, including its claims in view of its specification, the prosecution history of the '478 Patent, and the IPR2024-00078 Petition ("78 Petition") and its exhibits, including Petitioners' Expert's Declaration ("78 EX1003" or the "78 Expert Declaration").

27. I also reviewed the '348 Patent, including its claims in view of its specification, the prosecution history of the '348 Patent, and the IPR2024-00079 Petition (the "79 Petition") and its exhibits, including Petitioners' Expert's Declaration ("79 EX1003" or the "79 Expert Declaration").

28. In this Declaration, I may refer to the "78 Petition" and the "79 Petition" collectively as the "Petition" and may refer to the "78 Expert

Declaration" and the "79 Expert Declaration" collectively as the "Petitioners' Expert's Declaration" when referring to portions of the documents that are substantially the same between the proceedings.

E. Extrinsic Materials Considered

29. In preparing this Declaration, I also reviewed the following exhibits:

Exhibit	Description
EX2003	Computer Busses, Bill Buchanan, CRC Press, 2000, pp. 139-143
EX2004	The Multi-boot Configuration Handbook, Roderick W. Smith, Que, 2000, p. 71, 76-79
EX2005	Encyclopedia of Microcomputers, Volume 2 - Authoring Systems for Interactive Video to Compiler Design; Allen Kent, James G. Williams, and Rosalind Kent; Taylor & Francis; 1988; pp. 196-197
EX2006	PhoenixBIOS 4.0 Revision 6 User's Manual; Phoenix Technologies Ltd.; June 22, 2000; pp. ii-iv, 31-34, 39, 43-51, 80
EX2007	Forensic Computing: A Practitioner's Guide; A. J. Sammes, Brian Jenkinson; Springer Science & Business Media; 2000; pp. 119; 132, 145
EX2008	Introduction to Software Engineering; Ronald J. Leach; CRC-Press; 2000; p. 292
EX2009	PC Interrupts: A Programmer's Reference to BIOS, DOS, and Third-party Calls; Ralf Brown, James Kyle; Addison-Wesley; 1991; pp. 7-8, 205-211
EX2010	Hard Disk Controller: the Disk Drive's Brain and Body; James Jeppesen, Walt Allen, Steve Anderson, Michael Pilsl; Infineon Technologies; 2001
EX2011	SCSI-2 Standard; ANSI; 1994; pp. i-vi, 150-217
EX2012	ATA Draft Standard; ANSI; 1994; pp. i-iii, 33, 52-209
EX2013	Newton's Telecom Dictionary: The Official Dictionary of Telecommunications Networking and Internet; 16th Edition; Harry Newton, Ray Horak; Cmp Books; 2000; p. 398
EX2014	U.S. Patent Publication No. 2009/0254594 to Laurion D. Burchall, Microsoft, filed April 2, 2008
EX2015	*Reserved*

Exhibit	Description
EX2016	U.S. Patent Publication No. US20070226436 to Cheng et al., Microsoft, filed February 21, 2006
EX2017	U.S. Patent Publication No. US20040003103 to Witt et al., Microsoft, filed June 28, 2002
EX2018	Foundations of Computer Technology; A. John Anderson; 1994; Taylor & Francis; pp. 277, 282, 288-290

F. Legal Understanding

30. I am not a lawyer. In forming my opinions as set forth in this Declaration, I have relied upon legal principles supplied to me by attorneys for Datanet. In forming my opinions, I have not relied upon Datanet's attorneys for any purpose other than the following legal principles.

a. Claim Construction

31. I understand that determining whether a patent claim is valid requires a two-step analysis. First, the claim must be construed. Second, the properly construed claim must be compared to the prior art.

32. With respect to the first step, I understand that claims are construed from the perspective of a POSITA at the time of the purported invention. I understand that in the context of an *inter partes* review proceeding, absent a disclaimer or express definition, claim terms are given their plain and ordinary meaning to a POSITA in view of the specification and file history and the knowledge of a POSITA. Unless otherwise noted, I have applied what I consider

to be the plain and ordinary meaning of all terms as understood by a POSITA at the time of the priority date of each patent discussed above.

b. Anticipation

33. I understand that a patent claim is presumed to be valid. I understand that to overcome that presumption in an *inter partes* review proceeding, the challenger must show that the claim is invalid by a preponderance of the evidence. I understand the preponderance of the evidence to require the greater weight of evidence in favor of the challenger.

34. I understand that, although not expressly disclosed, subject matter may be inherently disclosed in a prior art reference where that subject matter is necessarily present in the subject matter disclosed and would be understood to be so by those of ordinary skill in the art. I understand that the fact that a certain result or characteristic may occur or be present in the prior art is not sufficient to establish the inherency of that result or characteristic.

35. I understand that a claim is invalid if each and every element of the claim is disclosed in a single prior art reference with the elements arranged as specified in the claim, either expressly or inherently. I understand that invalidity based on the disclosure of a single prior art reference in this manner is called anticipation.

c. Obviousness

36. I understand that a claim would have been obvious under 35 U.S.C. § 103 if one or more prior art references disclose, expressly or inherently, every claim limitation in such a way so as to render the claim, as a whole, obvious to a person of ordinary skill in the art at the time the purported invention was made. The relevant standard for obviousness is as follows:

> "A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made." 35 U.S.C. § 103 (pre-AIA)

37. In determining whether or not a patented invention would have been obvious, the following factors should be considered: (a) the scope and content of the prior art; (b) the differences between the prior art and the claims at issue; (c) the level of ordinary skill in the art; and (d) whatever "secondary considerations" may be present.

38. I understand that certain "secondary considerations" may be relevant in determining whether or not an invention would have been obvious, and that these secondary considerations may include commercial success of a product using the invention, if that commercial success is due to the invention; long-felt but

unsolved need for the invention; evidence of copying of the claimed invention by others; industry acceptance; initial skepticism of the invention; failure of others to solve the problem; praise of the invention by others; the taking of licenses under the patents by others; and simultaneous invention by others (which may be an indication that the claimed invention was the result of ordinary skill in the art).

39. I understand that a patent composed of several elements is not proved obvious merely by demonstrating that each of its elements was, independently, known in the prior art. While multiple prior art references or elements may, in some circumstances, be combined to render a patent claim obvious, I understand that I should consider whether there is an apparent reason to combine the prior art references or elements in the way the patent claims. To determine whether such an apparent reason exists to combine the prior art references or elements in the way a patent claims, it will often be necessary to look to the interrelated teaching of multiple patents, to the effects of demands known to the design community or present in the marketplace, and to the background knowledge possessed by a person having ordinary skill in the art.

40. I also understand that when the prior art teaches away from combining prior art references or certain known elements, discovery of a successful means of combining them is more likely to be non-obvious. A prior art reference may be said to teach away from a patent when a person of ordinary skill, upon reading the

reference, would be discouraged from following the path set out in the patent or would be led in a direction divergent from the path that was taken by the patent. Evidence showing there was no reasonable expectation of succeeding in combining references, that the proposed modification or combination of the prior art would change the principle of operation of the prior art invention, or that the suggested combination of references would require a substantial reconstruction and redesign of the elements may support a conclusion of nonobviousness.

41. I also understand that it is not permissible to use hindsight in assessing whether a claimed invention is obvious. Rather, I understand that, to assess obviousness, one must place oneself in the shoes of a person having ordinary skill in the relevant field of technology at the time the inventions were made who is trying to address the issues or solve the problems faced by the inventor and ignore the knowledge one currently now has of the inventions.

42. I also understand that there are numerous ways in which to articulate the legal standard for obviousness, including: (1) combining prior art elements according to known methods to yield predictable results, (2) simple substitution of one known element for another to obtain predictable results, (3) use of a known technique to improve similar devices (methods, or products) in the same way, (4) applying a known technique to a known device, method, or product ready for improvement to yield predictable results, (5) choosing from a finite number of

identified, predictable solutions with a reasonable expectation of success, (6) known work in one field of endeavor may prompt variations of the work for use in either the same field or a different one based on design incentives or other market forces if the variations are predictable to one of ordinary skill in the art, and (7) some teaching, suggestion, or motivation in the prior art that would have led one of ordinary skill to modify the prior art reference or to combine prior art reference teachings to arrive at the claimed invention.

d. Person of Ordinary Skill in the Art

43. I understand that the meaning of claim terms and a patent's validity are to be determined from the vantage point of one of ordinary skill in the art at the time of the invention. In determining who would be one of such ordinary skill, I understand it is appropriate to consider criteria such as: (a) the type of problems encountered in the art; (b) prior art solutions to those problems; (c) the rapidity with which innovations are made; (d) the sophistication of the technology; and (e) the education level of active workers in the field.

G. Claim Construction

44. Based on my review of the Petitions and Petitioners' Expert's Declaration's comments pertaining to claim construction, my understanding is that Petition and Petitioners' Expert's Declaration give claim terms their plain and ordinary meaning as would be understood by a POSITA on the earliest effective

filing date, in light of the specification and file history. 78 Petition, 13; 78 EX1003, 15; 79 Petition, 7; 79 EX1003, 17.

45. I have not been asked to provide opinions on claim construction for all terms in the ’348 and ’478 Patents (though I reserve the right to do so in the future). Unless otherwise noted, I have applied the plain and ordinary meaning of all terms that I address in this Declaration.

H. Declaration Citations

46. Unless otherwise indicated, in this Declaration, citations to sections or paragraphs of this Declaration are not intended to be limiting, do not imply that the cited sections or paragraphs are the only support, and do not imply that support is limited to said cited sections or paragraphs.

47. In addition, unless otherwise indicated, in this Declaration, citations to sections or paragraphs of this Declaration imply the cited sections or paragraphs along with any subsections or subparagraphs of said cited sections or paragraphs.

48. Additionally, unless otherwise indicated, in this Declaration, citations to sections or paragraphs of this Declaration imply incorporating therein (i.e., at the citing sections or paragraphs) by reference said cited sections or paragraphs along with any subsections or subparagraphs of said cited sections or paragraphs.

I. Right to Supplement

49. Petitioners' Expert's Declaration's presented opinions, arguments, references, and citations.

50. In this Declaration, I present opinions, arguments, references, and citations to establish that the challenged claims of the '478 and '348 Patents are not obvious in view of the alleged prior-art references presented in Petitioners' Expert's Declaration and taking into account the opinions, arguments, and references, and citations presented in Petitioners' Expert's Declaration.

51. In this Declaration, I do not attempt to address or rebut every opinion, argument, reference, or citation presented in Petitioners' Expert's Declaration.

52. My not addressing or not rebutting any opinion, argument, reference, or citation presented in Petitioners' Expert's Declaration should not be misconstrued as implying my agreement or my concession regarding any opinion, argument, reference, or citation presented in Petitioners' Expert's Declaration.

53. I reserve the right to modify and/or supplement my opinions, arguments, references, and citations and/or to present new ones for any reason and/or without any reason.

54. I also reserve the right to modify and/or supplement this Declaration, to present further analyses, and/or to amend my opinions as more and/or other

arguments and/or information becomes available as well as for any reason or for no reason.

J. Summary of Opinions

55. In the 78 Petition, Petition and Petitioners' Expert's Declaration challenge independent Claims 1 and 8-11 of the '478 Patent and challenge dependent Claims 2-6 of the '478 Patent, which depend from independent Claim 1. 78 Petition, 2; 78 EX1003, 17, 66

56. In the 79 Petition, Petition and Petitioners' Expert's Declaration challenge independent Claims 1, 8, and 12-15 of the '348 Patent; challenge dependent Claims 2-6, which depend from independent Claim 1; challenge dependent Claims 9-11, which depend from independent Claim 8; challenge dependent Claims 16, 18-20, 23, and 25-31, which depend from Claim 15; challenge dependent Claim 17, which depends from dependent Claim 16; and challenge dependent Claim 24, which depends from dependent Claim 23. 79 Petition, 2; 79 EX1003, 19, 78

57. Petition and Petitioners' Expert's Declaration present two sets of grounds under which Claims 1-6 and 8-11 of the '478 Patent and Claims 1-6, 8-20, and 23-31 of the '348 Patent are purportedly invalid. 78 Petition, 2; 78 EX1003, 17, 66; 79 Petition, 2; 79 EX1003, 19, 78

58. For both the ’478 Patent and the ’348 Patent, under Ground 1, Petition and Petitioners’ Expert’s Declaration proffer U.S. Patent No. 5,276,867 / EX1014 (“Kenley”) and U.S. Patent No. 5,325,519 / EX1004 (“Long”) as prior art references. 78 Petition, 2; 78 EX1003, 17; 79 Petition, 2; 79 EX1003, 19

59. For both the ’478 Patent and the ’348 Patent, under Ground 2, Petition and Petitioners’ Expert’s Declaration proffer U.S. Patent No. 5,953,729 / EX1005 (“Cabrera”), PCT Publication No. WO 99/12101 / EX1020 (“Schneider-PCT”), and U.S. Patent No. 5,778,395 / EX1006 (“Whiting”) as prior art references. 78 Petition, 2; 78 EX1003, 66; 79 Petition, 2; 79 EX1003, 78

60. In my opinion, for the reasons outlined below, Petition and Petitioners’ Expert’s Declaration have not established a reasonable likelihood of invalidating the challenged claims of the ’478 Patent and of the ’348 Patent as obvious.

61. Additionally, in my opinion, for the reasons outlined below, Claims 1-6 and 8-11 of the ’478 Patent and Claims 1-6, 8-20, and 23-31 of the ’348 Patent are not obvious in view of Kenley and Long and are not obvious in view of Cabrera, Schneider, and Whiting.

II. OVERVIEW OF THE ’478 AND ’348 PATENTS

62. I provide below a brief overview of features of the ’478 and ’348 Patents that are relevant to the arguments below.

63. The ’478 and ’348 Patents deal with file-system level files (rather than disk-drive level sectors) and operate at the file level (not sector level).

64. Each of the independent claims of the ’478 and ’348 Patents recites the word file, specifically the term “operating file” and the term “archive file.”

> “detecting an instruction by an operating system to perform an operation on an **operating file**;
> creating an **archive file** from the operating file and storing the archive file …” ’478, Claims 1, 8, 9, 10, 11; emphasis added

> “(a) the step of detecting an instruction by a resident program in a computing device for performing an operation on an **operating file**;
> (b) the step of creating an **archive file** from the operating file and storing the archive file …” ’348, Claims 1, 8, 12, 13, 14, 15; emphasis added

65. The Abstract of the ’478 and ’348 Patents describes “[a] method for **archiving files** includes determining when a change in an **operating file** is imminent, capturing the **operating file** immediately before the change in the **operating file** occurs, if the **operating file** has not already been captured; and capturing the **operating file** immediately after the change in the **operating file** has occurred.” Emphasis added

66. The word file(s) appears 196 times in the specification of each of the ’478 and ’348 Patents and appears 47 times in the claims of each of the ’478 and ’348 Patents.

67. The ’478 and ’348 Patents describe operations specific to files (not sectors), such as “open,” “close,” “rename,” and “delete.” ’478, 4:46-47, 6:54, 7:28, 7:57; ’348, 4:51-52, 6:59, 7:33, 7:62, 11:2-3

68. The words sector(s), track (other than “keep track”), head, and block (referring to storage unit) do not appear in the ’478 and ’348 Patents.

III. GROUND 1: CLAIMS 1-6 AND 8-11 OF THE ’478 PATENT AND CLAIMS 1-6, 8-20, AND 23-31 OF THE ’348 PATENT WOULD NOT HAVE BEEN OBVIOUS OVER KENLEY AND LONG.

69. In Ground 1, Petitioners’ Expert’s Declaration overlooks pivotal differences between the teachings of the ’478 and ’348 Patents on one hand versus the disclosures of Kenley and Long, individually and in combination, on the other hand.

70. Petitioners’ Expert’s Declaration neglects to address critical deficiencies in Kenley, in Long, and in the combination of Kenley and Long pertaining to the claims of the ’478 and ’348 Patents.

71. I describe these differences and deficiencies below.

A. Background

72. Petitioners’ Expert’s Declaration (¶¶ 59-95) relies on Long for most claim elements, except the database limitations and the file limitations.

73. As described below, Long and Kenley suffer from multiple deficiencies that make them unable to render obvious the challenged claims.

Neither reference (alone or in combination) could render the challenged claims obvious, and a POSITA would not have even been motivated to attempt to combine their elements.

74. Petitioners' Expert's Declaration uses Kenley for three main features in the alleged Kenley-Long combination towards the independent Claims of the '478 and '348 Patents: (a) file-based system, (b) creating and storing archive **files**, and (c) database. EX1003, ¶¶ 49-74.

75. I provide a brief overview of some relevant technical background below.

a. Data Backup and Management

76. In my data storage experience in the 1980s and 1990s, I witnessed the profound need for data backup and archival solutions.

77. As explained in the '478 and '348 Patents (1:15-2:55, 1:18-2:58) data protection and management has been a challenge for decades. In the 1990s, several factors heightened the vulnerability to data loss and, consequently, intensified the need for backup and archival solutions. These factors included the broadened popularity of home and office computers; the proliferation of computing devices, such as notebook computers and PDAs; the rapid and continual increase in data storage capacity; and the critical dependence on servers speared by the wide adoption of networking and the Internet.

78. Data loss can result from hardware failures, software failures, fires, stolen equipment, viruses, and user errors (accidental file deletions, file overwrites, errant programs, etc.).

79. Backup and archival solutions relied on data storage devices and subsystems, such as removable disk drives, flash drives, tape drives, optical drives, jukeboxes (AKA media changers or libraries; backup/archive devices that automatically load/unload removable storage media, such as tapes or optical disks), and disk arrays (RAID).

80. The backup device(s) may be local to the computer (such as a second disk drive or a removable flash drive embedded in the computer), external and on a local area network shared through a server, or remote and shared through wide area network.

81. Most commonly, the backup / archival was scheduled, manual, or based on mirroring or similar technology.

82. Two major classes of data backup and archival existed around the year 2000 and still exist today.

83. The first class of archival solution is sector (or block) based, where data of disk drive sectors is backed up on the same disk drive, on another disk drive, and/or on another type of storage device, such as tape drive or Jukebox.

84. With sector-based backup / archive systems, a major advantage is the ability to restore the entire disk drive if needed, for example if the local disk drive fails or is accidentally erased.

85. A major disadvantage of sector-based backup / archive is its unsuitability for recovering only a subset of the disk drive data, particularly files. Archiving and retrieving files requires that the archive software know how files are stored on the disk drive and how to access them. However, sector-based archival solutions operate at the sector level and rely on low-level device drivers or BIOS, which function at a lower level than the operating system and file system. In other words, there is no way to access files through low-level device drivers or BIOS, or even to distinguish file from non-file data on individual sectors.

86. The other class of archival solution is file based, where files are backed up on the same disk drive, on another disk drive, and/or on another type of storage device, such as tape drive or Jukebox.

87. With file-based backup / archive systems, a major advantage is the ability to restore individual selected files if needed, for example if a file is accidentally corrupted or deleted.

88. A major disadvantage of file-based backup / archive is the inability to restore the entire disk drive if needed, for example if the local disk drive fails or is accidentally erased. Archiving and retrieving sectors requires that the archive

software must read sectors. However, file-based archival solutions operate at the operating system and file system level and rely on the high-level operating system services. Non file-system data (such as the master boot record and the boot sector) can generally not be archived or restored with file-based solutions. EX2007, p. 145

89. In summary, sector-based archive solutions are different and incompatible with file-based archive solutions. Both solutions still exist separately today because of the difficulty in providing any sort of combined solution which can address the shortcomings inherent in each.

b. Disk Drive Sectors versus Files

i. Computer Data Storage Layers

90. Computer data storage follows a layered architecture, with each layer hiding the details of the next lower layer. EX1015, pp. 390-391

91. At the highest level are application programs, such as Microsoft Word, which allow the user to create, edit, and process files. Application programs call the operating system for system services, such as creation of files and accessing data storage services. The operating system (specifically the file system) manages the directories (folders) and files, including allocating space and providing access to directories and files. To access data storage devices, such as a disk drive, the operating system calls low-level device drivers (or BIOS), which

communicate with the data storage devices through interfaces, such as SCSI or IDE, using sector (or block) level commands, not files. In other words, file systems and disk sectors exist at separate conceptual levels in computer data storage architectures. File systems operate at the operating system level, are hardware agnostic (implementing the same functionality regardless of the type of data storage device used), and call hardware-specific disk drive device drivers to access the data stored in sectors (blocks). EX2005, p. 197; EX2018, pp. 277, 282, 288-290

92. For at least the above reasons, the techniques and solutions for managing data at the file system level are rarely applicable to the disk/sector level, and vice versa.

93. In a patent application filing of its own (EX2014), Petitioner Microsoft describes the well-known distinction between files and lower-level physical allocation units such as disk sectors.

> "Persistent data storage exists at several levels of abstraction, ranging from higher levels that are closer to the logical view of data seen by users running application programs, to lower levels that are closer to the underlying hardware that physically implements the storage. At a higher, logical level, data is most commonly stored as files residing in volumes or partitions, which are associated with one or more hard disks. The file system, which can be regarded as a component of the operating system executing on the computer, provides the interface between application programs and nonvolatile storage media, mapping the logically meaningful collection of data blocks in a file to their corresponding physical allocation units, or extents,

located on a storage medium, such as clusters or sectors on a magnetic disk." EX2014, ¶ 33

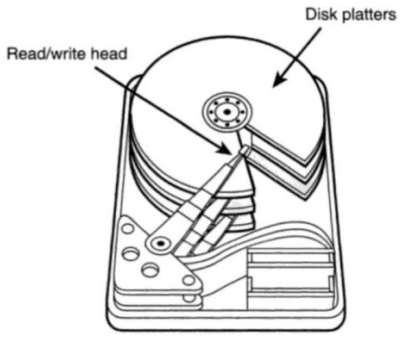
94. A POSITA would have known that computer data storage follows a layered architecture, with each layer hiding the details of the next lower layer.

ii. Disk Drive Sectors

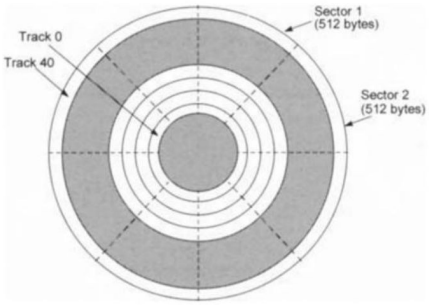
95. In this Declaration, I refer to sectors as pertaining to disk drives.

96. Disk drives are also called hard disk drives or hard drives to distinguish them from flexible disk drives or floppy drives.

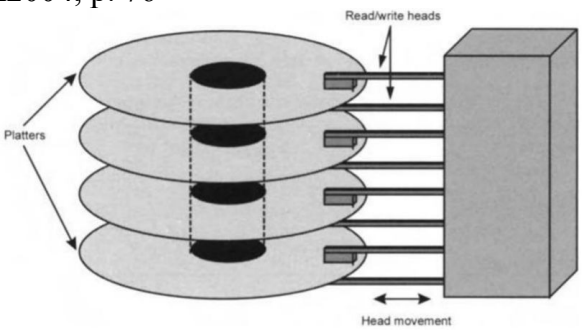
97. A disk drive consists of media platters coated with magnetic material. The surface of each platter is formatted to contain a series of concentric circles called tracks. Each track is segmented into and formatted to contain smaller data storage units called sectors. Each sector typically contains 512 bytes of data. For each platter surface, a dedicated head writes and reads data to and from sectors. The heads of a disk drive are constructed and affixed together as a head assembly, moving together, seeking to tracks, and writing and reading data to and from sectors. When the heads are at a particular location, the collection of tracks in that location is called a cylinder; in other words, a cylinder is a collection of tracks in a particular head position; the cylinder is given its name, because the track configuration or collection for any track location resembles a cylinder. EX2003, pp. 139-141, EX2004, p. 76



EX2004, p. 76



EX2003, pp. 140



EX2003, pp. 141

98. The term block (AKA logical block) refers to a unit of data transfer, typically one or multiple sectors.

99. Host computers exchange data with disk drives through data storage interfaces. In 2000, the two most common data storage interfaces were ATA (AKA IDE or EIDE) (AT Attachment; Integrated Drive Electronics; Enhanced IDE) and SCSI (Small Computer System Interface). EX2004, p. 76-79

100. Depending on the disk drive interface, read and write commands use sectors or blocks as units of data transfer. To read or write a disk drive sector, the read or write command specifies the sector location as cylinder-head-sector (CHS). Alternatively, to read or write a disk drive block (logical block), the read or write

command specifies the block location as logical block address (LBA). ATA used CHS addressing, but newer ATA versions also supported LBA addressing. SCSI used LBA addressing. EX2004, p. 76-79

iii. Files

101. In this Declaration, I apply the plain and ordinary meaning to the term file as controlled by the operating system, as a POSITA would have understood file to mean in the context of the '348 and '478 Patents.

102. Petitioners' Expert's Declaration at ¶ 37, citing its exhibit EX1012 (Midgdey), states that "[i]nformation processed by a computer is organized into distinct units called files, each of which contains related data."

103. Petitioners' Expert's Declaration at ¶ 37, citing its exhibit EX1015 (Silberschatz), states that "[t]he operating system provides system calls to create, write, read, reset, and delete files."

104. In other words, a file roughly consists of one or multiple sectors (or blocks) and is created and managed by the file system of the operating system. More precisely, most operating systems commonly allocate storage to files in units called clusters, and a cluster is one or more blocks (or sectors). EX1015, pp. 390-391

105. Files and sectors are different and should not be conflated.

106. Below are just a few of the main differences between files and sectors.

107. Files are accessible by the end user. A user can create, populate, read, rename, and delete files. In contrast, sectors are not accessible by the end user. A user cannot directly read or write sectors.

108. Files consist of clusters, which consist of sectors. In contrast, sectors are generally the smallest addressable unit in a disk drive.

109. Files are managed by the operating system, specifically the file system. In contrast, sectors are managed by low-level device drivers or BIOS (Basic Input-Output System).

110. Files are managed by the operating system, because files need to be usable with a wide variety of devices. Additionally, file formats allow files to operate on multiple operating systems. For example, text files, Microsoft Word files, Microsoft Excel files, and PDF files are usable on multiple operating systems. In contrast, sectors do not have such requirements, and an individual sector, block, or cluster from one disk drive would have no practical meaning or use to applications or operating systems running on another disk drive.

111. Files are hardware agnostic / independent. File contents are the same regardless of the underlying storage device where the file is stored. In contrast, sectors are hardware dependent, specifically hard disk drive dependent and storage interface dependent.

112. Files are inaccessible without the operating system and file system. In contrast, sectors are accessible through low-level device drivers or BIOS.

113. When the computer first boots, files are inaccessible, whereas sectors are.

114. Files are addressed using file names and paths (folder or directory location). EX1015 pp. 360-370. In contrast, sectors (or blocks) are addressed using CHS or LBA locations.

115. The objective of file-based archive solutions is to backup file system data (the individual files), whereas the objective of sector-based archive solutions is to backup disk drive data (sectors' data).

116. File-based archive solutions can retrieve one or multiple files but not an entire disk drive, whereas sector-based archive solutions can retrieve one or multiple sectors or an entire disk drive. Sector-based archives cannot retrieve individual files.

117. File-based archive solutions can only be used if the operating system is functioning, whereas sector-based archive solutions can be used to restore the entire system, including the operating system.

118. Additionally, in my experience, the engineering skills required to design and implement sector-based archive solutions would have been very different from those required to design and implement file-based archive solutions.

For example, the specific engineering skills for sector-based archive solutions would have included experience with data storage devices, such as disk drives, tape drives, and jukeboxes; experience with programming and interfacing with low-level device drivers and/or BIOS; and experience with data storage interfaces, such as SCSI and IDE. None of these skills are needed for file-based archive solutions.

119. For at least these reasons, sector-based archive solutions are very different from file-based archive solutions.

c. BIOS

120. BIOS (Basic Input / Output System) is software or firmware that provides low-level system services for operating systems and programs – operations such as power-on hardware initialization, system booting, and services for interfacing with hardware and devices (including disk drives). The BIOS services for interfacing with hardware and devices are available to the operating system and application programs.

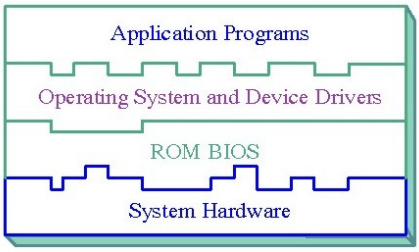
> BIOS provides routines for accessing I/O devices, such as the keyboard, screen, disk drives, and printer, attached to a microcomputer. This involves the following activities:
>
> - Checking that the device is ready
> - Initiating the data transfer
> - Checking for successful transfer
> - Performing error correction procedures, if necessary
> - Reporting successful completion or permanent error condition

EX2005, p. 196

A **ROM BIOS** (**Basic Input/Output System**) is a set of programs permanently stored in a **ROM** (Read-Only Memory) chip located on the computer motherboard. These programs micro-manage the hardware devices installed on your computer. When you turn on your computer, the ROM BIOS initializes and tests these devices. During run-time, the ROM BIOS provides the Operating System and application programs with access to these devices. You can also use the BIOS **Setup** program to change your computer's hardware or behavior.

Software works best when it operates in layers, and the ROM BIOS is the bottom-most software layer in the computer. It functions as the interface between the hardware and the other layers of software, isolating them from the details of how the hardware works. This arrangement enables you to change hardware devices without having to install a new operating system.

The following diagram shows the function of the ROM BIOS as the interface between the hardware and other layers of software:



EX2006, p. 31

The second task of the ROM BIOS is to provide the Operating System, device drivers, and application programs with access to the system hardware. It performs this task with a set of program routines called **BIOS Services**, which are loaded into high memory at boot time.

EX2006, p. 32

121. BIOS deals with disk drive sectors, not files. Files are handled by the operating system and its file system, but not by BIOS.

Most microcomputers are now equipped with disk drives and normally use a disk-based operating system (DOS). DOS routines supplement those in BIOS to provide extended services for using I/O devices. The division of responsibilities between BIOS and DOS for accessing disk storage devices is illustrated by the IBM PC:

- BIOS can read or write to a specified sector, track, and side of a diskette.
- DOS has responsibility for management of space on the diskette, including maintenance of a directory of the files and recording the space that has been allocated.
- DOS provides access to files and their records as logical entities and not by their physical location.

-33-

EX2005, p. 197

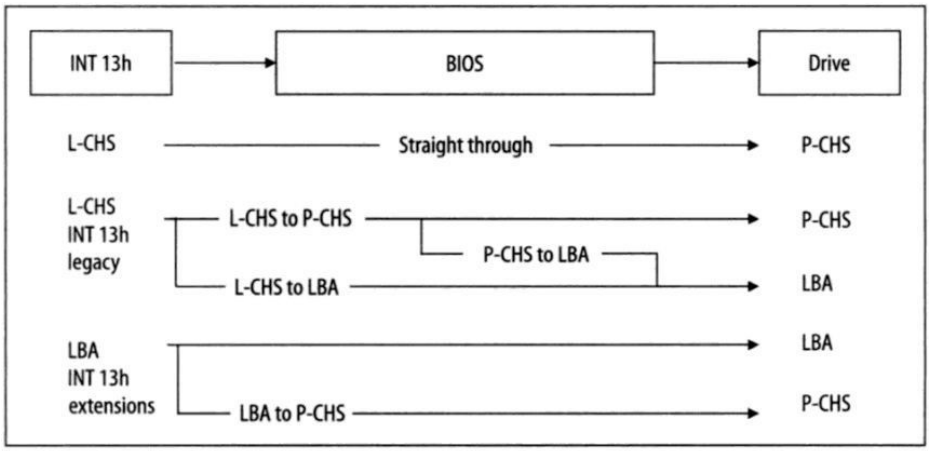


Fig. 5.20 Translation options.

EX2007, p. 132

122. Exhibit EX2006 p. 80 shows a list of BIOS services. There are no file-based services or functions. All the functions are sector (or block) based. In particular, interrupt 13h is "BIOS Fixed Disk/Diskette Interface." Exhibit EX2006 pp. 43-51 show the various functions available through interrupt 13h, all of which are sector (or block) based. Exhibit EX2006 p. 33 shows disk drive (AKA fixed disk or hard drive) cylinder-head-sector geometries.

123. In the 1990s and in 2000, operating system software was produced by vendors such as Microsoft and Apple, whereas BIOS software was produced by vendors such as AMI (American Megatrends, Inc.), Phoenix, and Award. EX2004, p. 71

124. As shown in Exhibit EX2006 p. 32, Phoenix BIOS was used in a variety of operating systems, such as DOS, OS/2, UNIX, Windows 95, and Windows NT. Most other BIOS vendors supported these operating systems.

125. In that timeframe, BIOS functions may be replaced by functions provided by device drivers or memory-resident programs offering similar services to those provided by BIOS.

126. In summary, BIOS, or an equivalent memory-resident program, deals with sector-based or block-based functions but not file-based functions

B. Long and Kenley Overview

127. Below, I provide a short overview of certain pertinent features of Long and Kenley. In arguing obviousness of the independent Claims of the ’478 and ’348 Patents under Ground 1, Petitioners’ Expert’s Declaration relies primarily on Long, but uses Kenley for three main features in the alleged combination of Kenley and Long: (a) file-based system, (b) creating and storing archive files, and (c) database as Kenley’s main features. EX1003, ¶¶ 49-74

a. Long Overview

128. Long is a sector-based backup and archival system with a mass storage device (e.g., disk drive). Long allocates a partition of the mass storage device for storing backup data and supplemental backup information. To perform backups, Long detects accesses to the mass storage device that would alter its data;

upon such detection, Long stores the data to be altered in the audit partition region for potential later restoration.

i. Long is based on disk drive sectors, not files.

129. Unlike the ’478 and ’348 Patents, which are directed to archiving and recovering files, Long is directed to archiving and recovering disk sector data.

130. Petitioners’ Expert’s Declaration at ¶ 53 alleges that “a POSITA would have known that Long is applicable to files.” I disagree.

131. As described earlier in the section titled *Disk Drive Sectors versus Files*, files and sectors are different. The ’478 and ’348 Patents are directed to, based on, and deal with files, not non-file specific sectors. In contrast, Long is directed to, based on, and deals with just disk drive sectors, not files.

132. The detailed explanation in the following sections show how deeply Long is sector based, not file based, and show how the sector-based approach is intrinsic to, fundamental to, and embedded in Long’s invention.

133. Long’s invention, operations, events, and functions deal with disk drive sectors, deal with disk drive tracks, and deal with storage devices at low level. Long’s implementation is done using BIOS or low-level device driver. Long’s invention does not deal with files, file systems, or operating systems.

134. Long is not applicable to files, especially as Petitioners’ Expert’s Declaration postulates. The combination contemplated in Petitioners’ Expert’s

Declaration relies on Long for most of the ’478 and ’348 claim limitations, other than the database. EX1003, ¶¶ 59-95

135. An understanding of how extensively sector based Long is would undoubtedly deter a POSITA from considering Long for an alleged implementation of the ’478 and ’348 Patents, let alone contemplating modifying Long to make it file based, especially if Long is to be relied on for most of the claim limitations of the ’478 and ’348 Patents, as Petitioners’ Expert’s Declaration contemplates.

136. A POSITA with the understanding of how extensively sector based Long is would have realized that modifying Long to make it file based, as Petitioners’ Expert’s Declaration hypothesizes, is tantamount to a new invention.

137. Long does not mention the word file(s). Long does not suggest or imply anything that a POSITA would interpret as files.

138. Long also does not mention operations specific to files, such as “open,” “close,” “rename,” and “delete.”

139. Instead, Long uses the word sector(s) 42 times.

140. Below are a few examples of Long’s reliance on sectors.

> “**READ SECTORS** FROM AUDIT BUFFER” and “**WRITE SECTORS** TO ARCHIVE MEDIA.” EX1004, FIG. 6 blocks 66 and 68; emphasis added
>
> “CLoc The physical location of the last data in the buffer. The next update of the buffer will begin at this address. This is on a

> **sector by sector basis**." *Id.*, Table 1 columns 5-6; emphasis added
>
> "This event occurs when the data on the hard disk(s) is changed. The event contains the data that existed on **one sector** of the disk prior to the write occurring." *Id.*, 6:46-48; emphasis added
>
> "[t]he data to be modified on the disk is **read one sector at a time** (block 50). **Each sector is read** from the disk (or from a disk cache, if provided) and is placed in the audit buffer as "before image" data (block 52). If the application requests a write that is more than one sector, **each sector is processed** as an independent request." *Id.*, 7:37-43; emphasis added
>
> "After all of **the sectors of the write request** have been processed, the audit subsystem returns control back to the processing unit 14 and allows it to modify the hard disks 18 with the requested write." *Id.*, 8:14-17; emphasis added

141. Additionally, Petitioners' Expert's Declaration at ¶ 53 concedes that "Long does not specifically consider data stored in files."

ii. Long handles sectors containing file data or non-file data but does so on sector basis.

142. Although Long is sector based, Long can process sectors containing file data as well as sectors containing non-file data, but Long does so on sector basis and with no regard to whether the data is file data or non-file data. In other words, Long can process sectors containing data belonging to and organized as files, and Long can process sectors containing data not belonging to and not organized as files, but Long cannot distinguish file from non-file sectors. Long

processes file data and non-file data identically, without even knowing if the data is file data and non-file data.

iii. Long discloses disk drive tracks.

143. Long also discloses dealing with disk drive tracks, which, as described earlier in the section titled *Disk Drive Sectors*, contain sectors.

> "When a five second ticker is enabled and the system has determined that it needs to archive, **one track's data** (the size of the track is dependent upon the structure of the archive media) is prepared and then stored on archive media." EX1004, 8:38-42; emphasis added

> "The archive end marks the end of the **archive track** that is to be written. The **track is then written** to the archive media in block 68." *Id.*, 9:4-7; emphasis added

144. Long's dealing with disk drive tracks further confirms that Long is sector-based.

iv. Long describes dealing with data storage devices, but not files and not file systems.

145. Long also describes dealing with data storage devices, but not files and not file systems, and also describes locations on storage devices and states of data storage devices, which are sector-level operations, not file-level operations.

> "Responsive to a user request, data is restored from the sequence of data transfers in the audit partition region to the respective set of **locations on the storage device** to return the storage device to a **previous state**." EX1004, 10:26-30; emphasis added

"The computer system of the present invention includes a **mass storage device** having a defined audit partition region." *Id.*, 1:65-67; emphasis added

"From the history, the user may select a time to which **the state of the hard disk** will be restored." *Id.*, 9:24-25; emphasis added

"The present invention is applicable to any **mass storage media** wherein old data may be overwritten with new data. While the mass media described herein is **described as a "disk"**, it should be noted that other structures, such as a drum structure, could similarly be used without affecting the applicability of the present invention. Further, it should be noted that while the mass storage media typically remains with the computer system 10, the present invention may be used with removable media as well." *Id.*, 9:54-63; emphasis added

"a **storage device** comprising … responsive to a user request, restoring data from the sequence of data transfers in the audit partition region to the respective set of **locations on the storage device** to return the storage device to a **previous state**." *Id.*, 10:12-30; emphasis added

"restoring data from said audit partition region to said **storage device** responsive to a user request to return the storage device to a **previous state**." *Id.*, 11:31-34; emphasis added

"responsive to a user request, restoring data from the audit partition region to the respective set of **locations on the storage device** to return the storage device to a **previous state**." *Id.*, 11:34-37; emphasis added

"responsive to a user input, restoring data from the archival memory to the respective set of **locations on the storage device** to return the state of the storage device to a **previous state**." *Id.*, 14:19-22; emphasis added

146. Long's descriptions of disk drives and storage devices further confirms that Long is sector-based.

v. Long's operations and events relate to data storage devices, not files.

147. The '478 and '348 Patents are directed to files and describe operations specific to files (not sectors), such as "open," "close," "rename," and "delete." In contrast, Long describes hardware and low-level operations and events that relate to data storage devices, not files – operations and events such booting, rebooting, and power failure.

> "The system events may include, but are not limited to, a **start** condition (for enabling the audit subsystem), **power on** and **power failure**, **reboot**, **quiesce** (a user invoked marker for indicating a desired hard disk state) and audit (data changes)." EX1004, 2:8-12; emphasis added
>
> "The audit buffer 30 is used to record **events such as writes to the hard disk(s), power failure, power on and system reboot**." *Id.*, 5:26-28; emphasis added
>
> "The audit partition may be used to store information relating to any number of events. The events supported in the preferred embodiment are described in connection with Table 4. The computer system 10 provides circuitry for generating the appropriate signals responsive to the events. For example, the power supply generates a signal when AC **power is lost**, even though the UPS will continue to operate the computer system 10. Also, the interrupt for a **system reboot** is captured and stored prior to performing the reboot." *Id.*, 6:27-36; emphasis added

148. Long's descriptions of its operations that are specific to disk drives and storage devices (but not operations on files and file systems) further confirms that Long is sector-based.

149. Long also teaches that its rollback utility program reboots the system after restoration is complete. A POSITA would have understood that such approaches are unusable in file-level implementations, which operate at the operating system and file system level.

> "Typically, **rollback is performed via a utility program** that accesses the on-line system.
> …
> At that point, the hard disk has been restored to the selected rollback point. After rollback is thus completed, and the user exits the utility, **the utility will reboot the operating system**. After the operating system is loaded, the hard disks will contain the data as of the time of the selected rollback point." *Id.*, 9:13-14, 9:45-50; emphasis added

150. Long teaches operating at the lowest level, namely hardware level. For example, Long teaches hardware-level fault tolerance and hardware-based error detection methods, such as timing checks, ECC, and parity.

> "The detection of an error or its symptoms during the normal operation of the system is the cornerstone to **fault tolerant** architectures. The component complexity affects the ability to distinguish errors from correct values. Errors occurring in **data-storage components, such as registers and memory**, or during **data transmission via buses or network links**, are more easily detected than errors originating within modules that generate or transform data. The following methods are used for error detection and/or masking:
> 1. Module replication for error detection and masking.

2. **Protocol and timing** checks.
3. Error detection and correction codes (**ECC**).
4. Error detection **parity** check codes.
5. Self-checking **logic** (i.e., voting logic)." *Id.*, 3:3-18; emphasis added

vi. Long is implemented as part of the BIOS or as a memory resident program.

151. Long is implemented either as part of the BIOS or as a memory resident program.

152. Long at 6:60-63 states that "[i]n the preferred embodiment, the audit subsystem is **part of the system BIOS** (Basic Input/Output System), but could also be a **memory-resident program**." Emphasis added

153. A POSITA would have also known that, in the context of Long, which is sector (not file) based, a "memory-resident program," similar to BIOS, deals with disk drive sectors, not files, and would have known that files are handled by the operating system and file system, not by such memory-resident program.

154. Long's memory-resident program should not be confused with the resident program of the '478 and '348 Patents.

155. The '478 and '348 Patents define resident program as an "operating system (OS) or other program that has control over file operations such as "read", "write", "save", "rename", "delete", "copy", "move", "open", "close", etc." '478, 4: 44-47; '348, 4:49-52

156. In contrast, as described earlier, Long deals with sectors, not files, and does not mention, suggest, or imply file operations such as "open," "close," "rename," or "delete."

157. In summary, BIOS, or an equivalent memory-resident program, deals with sector-based or block-based functions but not file-based functions.

vii. Long teaches detecting access requests to alter data on storage device locations (sectors) but not files.

158. Instead of disclosing detecting an instruction to perform an operation on an operating file, Long teaches detecting access requests to alter data on storage device locations (sectors) but not files.

> "A computer system comprises a storage device comprising a plurality of locations for storing data and having a defined audit partition region comprising one or more of said locations and processing circuitry for **detecting access requests to alter data on respective sets of one or more of locations of the storage device**. Responsive to each access request, the processing circuitry stores data from said respective set of locations in the audit partition region prior to performing the access request, such that a sequence of data transfers from the storage device is stored in the audit partition region in a known order. Responsive to a user request, data is restored from the sequence of data transfers in the audit partition region to the respective set of locations on the storage device to return the storage device to a previous state." EX1004, Abstract; emphasis added

> "Processing circuitry **detects accesses to the device which would alter data** thereon, and stores the **data to be altered** in the audit partition region for later restoration." *Id.*, 1:67-2:2; emphasis added

> "processing circuitry for: detecting access requests to alter data on respective sets of one or more of locations of the storage device" *Id.*, 10:16-19

> "processing circuitry for: detecting access requests to alter data on respective sets of one or more of said locations of the storage device" *Id.*, 11:20-24

viii. A POSITA would have understood Long as teaching away from file system implementation.

159. In light of all the Long teachings described above, a POSITA would have understood Long as teaching away from file system implementation, and a POSITA would have not been motivated to, would have been dissuaded from, and would not have considered incorporating Long's disk/sector level methods into any file-level implementation (or vice versa).

160. A POSITA would have known that a file-level implementation would necessarily need to be implemented at a much higher level than the "processing circuitry" or BIOS taught by Long.

161. A POSITA would have realized that incorporating Long's disk/sector level methods into a file-level implementation would destroy the file system's ability to remain hardware agnostic (EX2014, ¶ 33).

162. A POSITA seeking to improve a file-level implementation would not have considered a reference such as Long, which teaches the lowest possible level implementation, relying on hardware, BIOS, and detection circuitry. Indeed Long,

and its constituent components do not deal with or even recognize files or the concept of files.

ix. In summary, Long is sector based, not file based, and not applicable to files as contemplated in Petitioners' Expert's Declaration.

163. In summary, unlike the '478 and '348 Patents, which are directed to files, Long is directed to sectors. Long is deeply and intrinsically sector based.

164. Long is not applicable to files, especially as Petitioners' Expert's Declaration postulates, which relies on Long extensively and as purely file based.

165. Modifying Long to make it file based would have been a new invention, and not something that would have been obvious to a POSITA.

b. Kenley Overview

166. Kenley is a file-based backup and archival system. Kenley discloses a variation on the concept of hierarchical data storage system for files with a primary storage element, a secondary storage element, and a backing store element, each progressively with greater storage capacity and slower access speed. Data is migrated from to the backing storage elements as fullness threshold is reached or based on time. EX1014, 2:33-68

C. A POSITA would not have combined Kenley and Long.

167. As discussed above, a POSITA would have understood the significant barriers to any combination of Kenley and Long and, as a result, would not have been motivated to combine the two references.

a. Petitioner's references teach away from the combination of Kenley and Long.

i. Schneider-US and Schneider-PCT

168. Petitioners' Expert's Declaration cites Schneider-PCT (EX1020) as prior-art reference in Ground 2 but does not cite Schneider-PCT at all for Ground 1. 78 EX1003, p. 66; 79 EX1003, p. 89

169. Schneider-PCT (EX1020) and Schneider-US (EX1016) are similar and by the same inventors.

170. Throughout its specification, Schneider-PCT erroneously refers to Long as '5**7**9, instead of '5**1**9.

171. All the Long teach-aways in Schneider-US are present verbatim in Schneider-PCT.

172. In addition, as described later in Ground 2 in the section titled *Schneider-PCT Deficiency #1: Schneider-PCT teaches away from any file-level implementations*, Schneider-PCT recommends sector-level implementation and teaches away from file-level implementation.

173. Based on Schneider-PCT's teach-away admonitions against file-based solutions and based on Schneider-PCT's teach-away admonitions against Long, a POSITA would have been motivated to NOT modify Long to make it file based.

174. In other words, based on Schneider-PCT teachings, a POSITA would have been motivated to keep Long's sector-based implementation rather than attempt to convert Long to a file-based implementation.

175. Metaphorically, the Schneider-PCT teach-aways about Long are akin to a second negative review by the same reviewer about the same restaurant, in which the reviewer complains not only about slow service but also about unhealthy food. A reader of the review would not be motivated to try that restaurant, let alone frequent it.

176. Notably, Petition and Petitioners' Expert's Declaration chose Schneider-US rather than Schneider-PCT to support the flawed hypothesis that a POSITA would have modified Long to make it file based.

177. I find such choice not surprising considering that Schneider-PCT's teach-aways are even more detrimental to the hypothesis than Schneider-US's teach-aways.

Sector-level archival provides better protection than file-level archival.

178. According to Schneider-PCT and as described later in Ground 2 in the section titled *Schneider-PCT Deficiency #1: Schneider-PCT teaches away from*

any file-level implementations, sector-level protection protects file data and non-file data, offering better fault tolerance and redundancy protection and resilience than file-level protection.

Sector-level archival is operating system independent.

179. According to Schneider-PCT and as described later in Ground 2 in the section titled *Schneider-PCT Deficiency #1: Schneider-PCT teaches away from any file-level implementations*, sector-level archival is operating system independent, making sector-level implementation more versatile.

Schneider-PCT explicitly teaches that its sector-level implementation and its file-level implementation are different.

180. Schneider-PCT teaches that its sector-level implementation and its file-level implementation are different. Schneider-PCT acknowledges and describes the differences between its sector-level implementation and its file-level implementation.

> "Depending on what is saved either a **sector (disk) level implementation** of the present invention is used, or a **file (operating system) level implementation**. The **two implementations substantially differ in method** but both yield the same end results to the user according to the above Statement." EX1020, 140:21-24; emphasis added
>
> "In a file level implementation of the present invention, with tight integration into the operating system, **the method and understanding by the user of retrieving information may significantly differ, although is still based on the principal of saving original states of recent changes**. For example, an operating system may simply present the user with a list of

> prior states for a given file, and allow any to be copied forward. **This is <u>quite different than reverting a disk</u>**. The file level implementation of present invention automatically keeps backup copies of files **prior to their being changed** in a circular system. As new backup files are added and the amount of disk storage dedicated to these backup files approach a preset limit, older backup files would be automatically discarded. Of course, if there is available free space on disk (as is known to the filing system), this limit can be ignored." EX102, 24:7-17; emphasis

181. Schneider-PCT teaches that file-level implementation is difficult to implement, prove, and correct.

> "**Although the present invention at the file level is possible it may be more difficult to implement, prove correct, and isolate as a subsystem**. Such an implementation could save an entire prior file state just before a file is opened and the intention to modify is clear, or it could attempt to only save prior states of the portion of the file being modified." EX1020, 24:20-24; emphasis

b. A POSITA would not have had a reasonable expectation of success in combining the references.

182. A POSITA would not have had a reasonable expectation of success in combining Kenley and Long for several reasons.

i. Long's teachings specific to sector-level implementation could not be reasonably expected to succeed in file-level implementation and would have contradicted Long's basic teachings.

183. A POSITA in 2000 would have understood that Long's teachings were specific to a sector-level implementation and could not be reasonably expected to succeed in a file-level implementation, and vice versa. Long was

explicitly designed to be sector based, not file based. Long does not mention, suggestion, or imply any file-based application for its invention.

184. A POSITA in 2000 would have understood that modifying Long into a file-based system or combining Long into a file-based system would have contradicted Long's basic and fundamental sector-based teachings, rendering Long inoperable for Long's intended purpose.

185. Furthermore, because of the well-known and fundamental differences between the file layer and sector layer described earlier in the section titled *Computer Data Storage Layers*, a POSITA in 2000 would have understood that extracting functionality from Long into Kenley (or vice versa) would have required substantial redesign, reconstruction, and rewrite of Long's extracted functionality. The well-established computer software hierarchy of layers is intended to separate the system functionality into levels with different capabilities. A POSITA in 2000 would have known that juxtaposing functions from different layers (higher into lower or lower into higher) would have involved significant redesign, reconstruction, and rewrite of the extracted functions.

186. For example, Long (EX1004 at 5:26-30) detects various types of events, such as "writes to the hard disk(s), power failure, power on and system reboot." These events occur at much lower layer than the operating system and file system layer. Detecting such events is inappropriate and possibly

unimplementable at the file level. Additionally, detecting such events at the operating system or file system level would change and negate the principle of operation of the file system and of the layered data storage architecture by embedding low-level functionality into a higher layer.

ii. Long is implemented as part of BIOS or resident program, both sector-based, low-level.

187. As described earlier the section titled *Long is implemented as part of the BIOS or as a memory resident program*, Long is implemented as part of BIOS or as resident program. Both are sector-based, low-level, unfitting, and inappropriate for a file-based solution. As described earlier in the section titled *Computer Data Storage Layers*, BIOS and the low-level memory-resident program (device driver) are below the operating system layer and do not deal with files. It would be a violation of computer data storage layered architecture for BIOS and the low-level disk drive device drivers to attempt to access the higher level file system of the operating system.

iii. Petitioners' Expert's Declaration's proposed Kenley-Long combination would have required substantial redesign and reconstruction of Long.

188. A POSITA would have known that modifying Long's system from sector based to file based, as suggested in the Petitioners' Expert's Declaration, is not trivial. Such endeavor would have entailed almost a complete redesign and reconstruction of Long, almost certainly necessitating a full rewrite of Long,

resulting in significant cost, risk, and time. Beyond such major redesign and reconstruction, Long and Kenley would still need to be integrated, further increasing the cost, risk, and time. The Kenley-Long combination contemplated in the Petitioners' Expert's Declaration requires extensive integration, affecting most aspects of Long. Integration alone (excluding the redesign, reconstruction, and rewrite) is not a trivial task.

> The second major topic covered in this chapter is software integration. Integration is the process of combining individual software modules and subsystems into a single working product. Software integration is not a trivial task, especially if the individual software units were developed in different locations or even by different companies separated geographically. This coordinated situation is typical in modern software development.
> EX2008, p. 292

189. Assuming, arguendo, that a POSITA considered Kenley and Long in combination to teach the individual elements of the '478 and '348 Patents, given the obvious and expected challenges of combining Kenley-Long as contemplated in Petitioners' Expert's Declaration, the POSITA would not have been motivated to combine Kenley and Long.

iv. Schneider-PCT warns that file-level implementation is difficult to implement, prove, and correct, hence teaching no reasonable expectation of success.

190. Petitioners' own references, Schneider-PCT and Schneider-US, warn that file-level implementation is difficult to implement, prove, and correct, which a POSITA would have understood as teach no reasonable expectation of success.

> "**Although the present invention at the file level is possible it may be more difficult to implement, prove correct, and isolate as a subsystem**. Such an implementation could save an entire prior file state just before a file is opened and the intention to modify is clear, or it could attempt to only save prior states of the portion of the file being modified." EX1020, 24:20-24; emphasis

v. Petitioner's patent application (EX2017) describes the difficulty of using sector-based solutions to retrieve files.

191. In a patent application filing of its own (EX2017), Petitioner Microsoft discloses a system for managing images, which are sector-by-sector copies of data contained in data storage media.

> "A method and system for **managing image files** is provided. The image files include a plurality of streams such as a control stream, a data stream, a bitmap stream, and a cluster map stream. An audit trail stream, properties stream and fix-up stream may also be provided. An image driver translates requests from the operating system so that the image can be read, edited or otherwise manipulated." EX2017, Abstract; emphasis added

> "An image file is a copy of the data stored on a source storage media volume. Typically, **the image file is a single stream of data that is a sector by sector copy of the data contained on the source storage media volume**. The image file is in turn stored on a destination storage media volume. The destination storage media volume is a different volume or partition on the same storage media or, alternatively, a separate storage media. For example, an image file that represents a sector by sector copy of a hard disk volume is stored on a CD ROM." *Id.*, ¶ 7; emphasis added

192. In the same patent application (EX2017), Petitioner Microsoft warns about the difficulty that the operating system cannot process the sector image data (sector backup) in order to retrieve individual files. A POSITA would have understood such warning as recognition that there was no reasonable expectation of success in combining a file-level reference with sector-level reference.

> "Because the **image file is created as a sector by sector** copy of the source storage media, and is stored as single data stream, **it is difficult to manipulate the image file**. The **operating system cannot readily read** the control information or individual file data from the data stream. As a result, **files in the data stream cannot be replaced with new files or edited**. In order to update an image file, it is necessary to restore the image file to a storage media volume, perform updates to the storage media volume, and then create a new image file." *Id.*, ¶ 7; emphasis added

193. Notably, this Microsoft patent application (EX2017) was filed in 2002 (after the filing of the '478 Patent). EX2017, cover page

194. Therefore, at least almost two years after the priority date of the '478 Patent, a POSITA would have understood that it would have been difficult to extract or process files contained within a sector-by-sector disk image (as contemplated in Petitioners' own proposed combinations). In other words, at the time of invention of the '478 Patent, a POSITA would have no reasonable expectation of success of combining a sector-based solution with a file-based solution.

vi. A POSITA would have recognized that the proposed modifications would have changed the principle of operation of the modified reference.

195. A POSITA would have understood that Long's invention is deeply and intrinsically sector based and would have recognized that the proposed modifications would have changed the principle of operation of the modified reference.

196. Long is sector based, not file based, and Long's invention is intrinsically sector based. Long's invention, operations, events, and functions deal with disk drive sectors, deal with disk drive tracks, and deal with storage devices at low level. Long's implementation is done using BIOS or low-level device driver. Long's invention does not deal with files, file systems, or operating systems. A POSITA would have realized that modifying Long to make it sector based is complicated, unlikely to succeed, and costly (even today).

197. A POSITA with the understanding of how extensively sector based Long is would have realized that modifying Long to make it file based is tantamount to a new invention (and vice versa).

198. Petitioners' Expert's Declaration at ¶ 25 states that "I understand that in considering obviousness, it is important not to determine obviousness using the benefit of hindsight derived from the descriptions found in the patent being considered."

199. In my opinion, the argument that a POSITA would have been motivated to modify Long to make it file based to be used in combination with Kenley is purely "hindsight derived from the descriptions found in the patent being considered." I am aware of nothing prior to the priority date of the '478 and '348 Patents that would have motivated a POSITA to make the proposed combination.

c. A POSITA would have recognized that the Petitioners' Expert's Declaration's proposed reasons to combine the references were deficient.

200. For the reasons described in the following sections, a POSITA would have recognized that the Petitioners' proposed reasons to combine the references were deficient.

i. Petitioners' Expert's Declaration's Schneider-US citations do not show Long operating at file level.

201. Petitioners' Expert's Declaration citations to Schneider-US do not show Long operating at file level.

202. Petitioners' Expert's Declaration at ¶ 57 conflates the Move Method ideas from Long and the Move Method supplemental features from Schneider-US.

203. Petitioners' Expert's Declaration at ¶ 57 states that Schneider-US "expressly describes use of Long's audit buffer technique for saving a file's data before the file is altered. EX1016, 46:36-47:8 (using file overwrite operation to explain "what normally happens when data is written" and then describing how the

Long technique (referred to as the "Move" method) saves a file just prior to it being overwritten)[.]"

204. The above Petitioners' Expert's Declaration statement is a mischaracterization of Schneider-US's disclosure.

205. As I explain below, Schneider-US (EX1016) at 46:36-47:8 does not "expressly describe[] use of Long's audit buffer technique for saving a file's data before the file is altered."

Schneider-US 46:36-47:8 (first part: 46:36-62)

206. Schneider-US (EX1016) at 46:36-62 describes details about FIG. 37.

207. As Schneider-US (at 5:56) describes, "FIG. 37 illustrates a normal write operation."

208. In other words, 46:36-62 describes how Schneider-US write operation functions generally but does not relate specifically to Long.

Schneider-US 46:36-47:8 (second part: 46:63-47:8) and FIG. 38

209. Schneider-US (EX1016) at 46:63-47:8 describes FIG. 38.

210. As Schneider-US (at 5:57-58) describes, "FIG. 38 illustrates the Move Method of writing data to a disk."

211. However, Schneider-US (EX1016) at 46:63-47:8 describes the Move Method including supplemental features from Schneider-US, such as handling fragmented files and delaying writes. Long does not disclose, suggest, or imply

any of these features. Indeed these features are from Schneider-US, not from Long and should not be attributed to Long.

212. In citing Schneider-US 46:63-47:8, Petitioners' Expert's Declaration at ¶ 57 conflates the Move Method ideas from Long and the Move Method supplemental features from Schneider-US. Petitioners' Expert's Declaration's conflation resulted in flawed analysis, which resulted in faulty conclusions.

213. Simply, these excerpts describe Schneider-US, not Long features. These features do not disclose that Long is operating at file level.

Schneider-US 46:25-26

214. Petitioners' Expert's Declaration at ¶ 57 cites Schneider-US at 46:25-26 as discussing "fundamental methods for saving the prior states of overwritten data.

215. This Schneider-US citation does not mention Long or the Move Method and does not disclose that Long is operating at file level.

Schneider-US 6:33-36

216. Petitioners' Expert's Declaration at ¶ 57 cites Schneider-US at 6:33-36 as presenting "fundamental methods of recording the original state of information prior to its being altered."

217. This Schneider-US citation does not mention Long or the Move Method and does not disclose that Long is operating at file level.

218. In fact, the following sentence teaches away from files by stating that "[t]he first four methods work substantially **outside of the OS's method of organizing and assigning its file to disk pages**." Schneider-US, 6:36-38; emphasis added

219. In summary, Petitioners' Expert's Declaration citations to Schneider-US do not disclose that Long is modified to make it file based or is operating at file level; therefore, Schneider-US does not confirm "reasonable expectation of success, and overall obviousness, of implementing Long's technique for a file-based system such as Kenley's[.]"

220. Schneider-US's use of Long does not preclude Long from operating as intended – at the sector level, not file level. In other words, Schneider-US uses Long as is, sector based, not file based.

221. There is nothing in Schneider-US to suggest that Long cannot be or is not used as designed and intended, at the sector level, not file level.

222. There is nothing in Schneider-US to suggest that Long should be or is modified to operate at file level, not sector level.

223. Furthermore, Schneider-US describes that the device driver is below the operating system, which is consistent with detecting at the sector level, not file level.

ii. **Long can and does operate at the sector level to perform the functions contemplated by Schneider-US within Schneider-US.**

> "The Driver could execute in the main processor with the external disk on a similar bus to the internal disk. In this case **the Driver directly controls the transferring of information to and from the disk**. An **alternate implementation incorporates the Driver into the external disk controller**. Here, the Driver receives requests through the disk interface. The difference between these two cases is in which side of the disk interface lies the Driver. This is illustrated in FIG. 59." EX1016, 66:25-32; emphasis added

224. Schneider-US FIG. 59 is reproduced below.

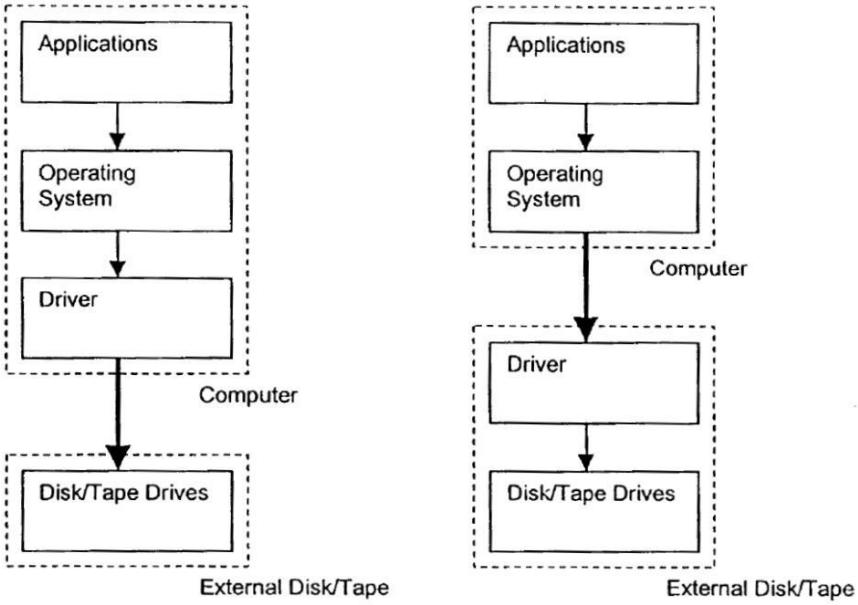


Figure 59

225. For at least the above reasons, within Schneider-US, Long can and does operate <u>at sector level</u> to perform the functions contemplated by Schneider-US at sector level.

226. In summary, Schneider-US's recitations of Long do not disclose that Long is NOT operating at sector level; therefore, Schneider-US does not confirm "reasonable expectation of success, and overall obviousness, of implementing Long's technique for a file-based system such as Kenley's[.]"

iii. **Schneider-US teaches away from Long and the Move Method as slow.**

227. Each of the independent claims of the '478 and '348 Patents recite a "temporally proximate" claim limitation.

228. The specifications of the '478 and '348 Patents also stress the importance of fast performance, for example by capturing files "immediately" and providing "near real time file capture," "minimal performance impact," and "imperceptible impact." '478, Abstract, 2:57-58, 3:12-13, 5:7-14, 7:7-10, 7:15-19, 7:45-46, 7:61-63, 8:11-15, 8:26, FIGURE 3, FIGURE 4, FIGURE 5; '348, Abstract, 2:59-60, 3:14-15, 5:12-19, 7:12-15, 7:20-24, 7:50-51, 7:66-8:1, 8:16-20, 8:21, FIGURE 3, FIGURE 4, FIGURE 5

229. Yet, Schneider-US repeatedly teaches away from Long by describing Long and the Move Method as slow, introducing delays, causing performance suffering, causing problematic overhead, and sacrificing performance.

<u>Teach-Away Example 1</u>

230. Schneider-US at 4:27-44 states that "Pat. No. 5,325,519, entitled "Fault Tolerant Computer with Archival 0 Rollback Capabilities", to **Long** et al. ("'519 patent"). The **'519 patent** discloses a storage device which includes processing circuitry for detecting access requests to alter data in respective locations of a storage device, and, prior to executing such requests, storing the data in such locations in a audit partition region of the storage device. The device of the '519 patent can subsequently restore the data retained in the audit partition region to its previous location on the device, and thereby return the storage device to a previous state. **The device and approach of the '519 patent, however, <u>inherently introduces delays</u> in writing data to the storage device. In some cases, these delays may make it <u>infeasible</u> to use this technology**. Therefore, there remains a need for a more fast, flexible and dynamic way to retain historical information in a computer system." Emphasis added

<u>Teach-Away Example 2</u>

231. Schneider-US at 6:62-7:4 warns that "[t]he **Move Method** involves first reading data about to be overwritten and saving it in a disk-based history buffer. It has the **drawback of fundamentally being slow**. … The next three methods offer better solutions to the throughput problem." Emphasis added

<u>Teach-Away Example 3</u>

232. Schneider-US at 8:67-9:1 warns that "the system's **performance suffers** in that the **Move Method** must be used." Emphasis added

Teach-Away Example 4

233. Schneider-US at 9:13-17 warns that "[a]n important aspect of this technique is using re-mapping techniques to allow the placing of data in alternate locations without having to fall back to **the Move Method and its problematic overhead**." Emphasis added

Teach-Away Example 5

234. Schneider-US at 9:32-33 warns that "[i]f an overflow occurs the Move Method is forced into **the slow Move Method**." Emphasis added

Teach-Away Example 6

235. Schneider-US at 9:36-39 warns that "with enough writes, **the Move Method's buffering can overflow**, whereas the Temp Method always has some alternate location to which to write new data." Emphasis added

Teach-Away Example 7

236. Schneider-US at 48:14-16 states that, of the five methods including the Move Method, "[i]n summary, **the Always and File Methods yield the best overall performance** by sacrificing some disk space in mapping overhead." Emphasis added

Teach-Away Example 8

237. Schneider-US at 48:32-34 warns that "[t]he **Move Method** is straightforward but in its simplicity, it **sacrifices the disk performance** to which users are accustomed." Emphasis added

238. Based solely on these statements from Schneider-US regarding Long, a POSITA would have been motivated to NOT use Long even as sector-based implementation and would have been motivated to NOT modify Long for purported file-based implementations.

239. The fact that Schneider-US uses Long is not an endorsement of Long, especially considering Schneider-US's teach-aways of Long.

240. Metaphorically, these Schneider-US teach-aways about Long are akin to a negative restaurant review in which the reviewer complains about slow service. A reader of the review would not be motivated to try that restaurant.

iv. Schneider-US teaches that the Move Method is outside the operating system.

241. Schneider-US also teaches that the Move Method is outside the operating system, which is consistent with the teachings of Long (sector based, not file based) unmodified. Files are handled by the operating system and its file system.

> "This application describes **five software methods** for information recovery in which the backup (historic) data is kept on the same hard disk as normally used by the user.
> …

> A computer's operating system (OS) typically stores information on a hard disk. The example embodiments of the present invention present **five fundamental methods** of recording the original state of information prior to its being altered. **The first four methods work substantially outside of the OS's method of organizing and assigning its file to disk pages**. They substantially differ in performance and how they utilize the disk. The last method calls for integrating the process of saving and retrieving original states of altered information directly into the OS's filing system.
> **1. Move Method: Move before overwrite**
> 2. Divert Method: Divert and later swap into place during free time
> 3. Temp Method: Temporarily re-map, swap into place during free time
> 4. Always Method: Always re-mapped, re-organize during free time
> 5. File Method: Implemented in the filing system at the file or portion of file level" Schneider-US, 6:20-50; emphasis added

242. In summary, Schneider-US discloses that the Move Method is outside the operating system, which is consistent with the teachings of Long as sector based, not file based; therefore, Schneider-US does not confirm "reasonable expectation of success, and overall obviousness, of implementing Long's technique for a file-based system such as Kenley's[.]"

v. Petitioners' Expert's Declaration's EX1013, EX1012, EX1010, and Kenley references do not cure Long's deficiencies related to its being sector based or its lack of detection.

243. Petitioners' Expert's Declaration at ¶ 53 states that "[t]he POSITA knew that data on a hard disk was most commonly organized as files to facilitate data access using a file system."

244. The fact that disk drives may contain data organized as files does not cure Long's deficiencies related to its being directed to sectors, including the deficiency that Long is sector based, not file based; the deficiency that Long does not disclose operating file; the deficiency that Long does not disclose archive file; the deficiency that Long is part of BIOS or a memory-resident program, which are sector based; the deficiency that Long does not disclose an instruction to perform an operation on an operating file; and the deficiency that Long does not disclose detecting an instruction to perform an operation on an operating file.

245. Files were known well before Long's invention, yet Long's inventors explicitly chose a sector-based solution, not a file-based solution.

246. To support Petitioners' Expert's Declaration's hypothesis of using Long for "implementing Long's technique for a file-based system," Petitioners' Expert's Declaration at ¶ 53 cites to EX1013, EX1012, EX1010, and Kenley.

247. However, Petitioners' Expert's Declaration's EX1013, EX1012, EX1010, and Kenley references do not cure Long's deficiencies related to its being sector based and do not suggest or confirm reasonable expectation of success or overall obviousness of implementing Long's technique for a file-based system.

248. I address EX1013, EX1012, EX1010, and Kenley below.

vi. EX1013 does not cure Long's deficiencies related to its being sector based or its lack of detection.

249. Petitioners' Expert's Declaration at ¶ 53 cites to EX1013 as "describing the File Allocation Table (FAT) file system created for the MS-DOS operating system[.]"

250. The fact that MS-DOS creates a FAT file system does not cure Long's deficiencies related to its being directed to sector, including the deficiency that Long is sector based, not file based; the deficiency that Long does not disclose operating file; the deficiency that Long does not disclose archive file; the deficiency that Long is part of BIOS or a memory-resident program; the deficiency that Long does not disclose an instruction to perform an operation on an operating file; and the deficiency that Long does not disclose detecting an instruction to perform an operation on an operating file.

251. The fact that MS-DOS creates a FAT file system does not make modifying Long to make it file based obvious.

252. The EX1013 citation also specifies MS-DOS. However, as described earlier in the section titled *Long is implemented as part of the BIOS or as a memory resident program*, MS-DOS BIOS (or an equivalent resident program) is strictly sector based and does not operate on files.

253. Additionally, as explained earlier in the section titled *Long handles sectors containing file data or non-file data but does so on sector basis*, Long is

not limited to sectors containing data belonging to files, and Long can process sectors containing data belonging to files or not. Long does so on sector basis.

254. For at least all the above reasons, EX1013 does not make modifying Long to make it file based obvious.

vii. EX1012 does not cure Long's deficiencies related to its being sector based or its lack of detection.

255. Petitioners' Expert's Declaration at ¶ 53 cites to EX1012 as "explaining different file systems for different operating systems such as MS-DOS, Macintosh, and UNIX … Indeed, the *background* section of one background reference flatly states what was well-known: "[i]nformation processed by a computer is organized into distinct units called files, each of which contains related data.""

256. The fact that different file systems exist for different operating systems and the fact that information processed by computers is organized as files do not cure Long's deficiencies related to its being directed to sector, including the deficiency that Long is sector based, not file based; the deficiency that Long does not disclose operating file; the *deficiency* that Long does not disclose archive file; the deficiency that Long is part of BIOS or a memory-resident program; the deficiency that Long does not disclose an instruction to perform an operation on an operating file; and the deficiency that Long does not disclose detecting an instruction to perform an operation on an operating file.

257. The fact that different file systems exist for different operating systems and the fact that information processed by computers is organized as files do not make modifying Long to make it file based obvious.

258. Additionally, as explained earlier in the section titled *Long handles sectors containing file data or non-file data but does so on sector basis*, Long is not limited to sectors containing data belonging files, and Long can process sectors containing data belonging to files or not. Long does so on sector basis.

259. For at least all the above reasons, EX1012 does not make modifying Long to make it file based obvious.

viii. EX1010 does not cure Long's deficiencies related to its being sector based or its lack of detection.

260. Petitioners' Expert's Declaration at ¶ 53 cites to EX1010, stating that "it was well-known that applications in file-based systems would perform write and other operations that modified the file."

261. The fact that applications can write to and modify files does not cure Long's deficiencies related to its being directed to sector, including the deficiency that Long is sector based, not file based; the deficiency that Long does not disclose operating file; the deficiency that Long does not disclose archive file; the deficiency that Long is part of BIOS or a memory-resident program; the deficiency that Long does not disclose an instruction to perform an operation on an operating

file; and the deficiency that Long does not disclose detecting an instruction to perform an operation on an operating file.

262. The fact that applications can writes and modify files does not make modifying Long to make it file based obvious.

263. Additionally, as explained earlier in the section titled *Long handles sectors containing file data or non-file data but does so on sector basis*, Long is not limited to sectors containing data belonging files, and Long can process sectors containing data belonging to files or not. Long does so on sector basis.

264. For at least all the above reasons, EX1012 does not make modifying Long to make it file based obvious.

ix. Kenley does not cure Long's deficiencies related to its being sector based or its lack of detection.

265. Petitioners' Expert's Declaration at ¶ 53 cites to Kenley, as explaining ""stor[ing] data organized as files" and using file attributes to track the backup location of files enables the data to be more easily and transparently recovered from archive storage media for use by an application. … (explaining how the file-based approach enables the system to "properly backup and restore files")."

266. The above Kenley statement does not cure Long's deficiencies related to its being directed to sectors, including the deficiency that Long is sector based, not file based; the deficiency that Long does not disclose operating file; the deficiency that Long does not disclose archive file; the deficiency that Long is part

of BIOS or a memory-resident program; the deficiency that Long does not disclose an instruction to perform an operation on an operating file; and the deficiency that Long does not disclose detecting an instruction to perform an operation on an operating file.

267. The above Kenley statement does not make modifying Long to make it file based obvious.

268. Additionally, as explained earlier in the section titled *Long handles sectors containing file data or non-file data but does so on sector basis*, Long is not limited to sectors containing data belonging files, and Long can process sectors containing data belonging to files or not. Long does so on sector basis.

269. For at least all the above reasons, Kenley would not have made it obvious to modify Long to make it file based.

x. Petitioner's patent application (EX2016) describes sector-level implementations and sector-level implementations as different and incompatible.

270. In a patent application filing of its own (EX2016), Petitioner Microsoft describes in the background section that "sector based" "disk imaging" and "file backup" solutions were known in the art and both were still widely used at the time of filing (February 21, 2006), which was well after the time that Petitioners contend that it would have been obvious to combine the two solutions.[20070226436 ¶¶ 1-2].

> "[0001] **Disk imaging is widely used to backup and restore data** on computer storage media. Disk imaging, as known in the art, captures a snapshot of the data on a computer hard disk. **The captured image may be used subsequently to re-create the disk**. **Existing disk imaging solutions are <u>sector based</u>**. That is, these solutions copy the disk physical data as it is laid out on the hard disk **sector by sector**, in raw binary format, **regardless of whether the data is useful or not** (e.g., even empty space is captured). Further, files that should not be captured, such as page files, are also captured resulting in large image files that inefficiently use storage resources. **Existing systems that store the large image files** on storage resources accessible over a network also inefficiently use network resources.
>
> [0002] Existing **file backup solutions save the logical content of the file system** in online mode (e.g., while executing an operating system **stored at least in part on the file system**). However, files such as some subset of operating system files **cannot be captured while the file system is online**. Further, the existing file backup solutions **do not preserve information relating to the disk partition, boot sector, and the like**. As such, the existing file backup solutions **cannot restore an entire disk to a prior state**." EX2016, ¶ 1-2; emphasis added

271. A POSITA would have understood that both solutions were (and are) still in use at that time (whether in 2000 or 2006), because each approach encompassed limitations inherent to their use.

272. For example, sector-based solutions copy the disk physical data sector by sector regardless of whether the data is useful or not, resulting in large and inefficient images. In contrast, file-based solutions cannot capture subsets of operating system files while the file system is online and also do not preserve disk partition information and boot sector data.

273. A POSITA would have understood that such limitations would have prevented any obvious combination.

274. In the same patent application filing (EX2016), Petitioner Microsoft purports to have invented a method of selectively capturing and restoring logical volumes by imaging the volumes while the volumes are offline and by including operating system files as well as volume and partition information, to facilitate restoring volumes to prior states.

> "[0004] Embodiments of the invention **capture and restore one or more logical volumes** of data on computer readable media. In an embodiment, the invention **images the volumes while the volumes are offline to include files associated with an operating system stored on the volumes**. Aspects of the invention further **capture volume and partition information regarding the physical layout of the volumes** such that the volumes may be restored to a state that is logically equivalent to a prior state. Further, the system supports **capture of selected data**. The system also enables efficient and effective defragmentation of the volumes." *Id.*, 0004; emphasis added

275. Contrary to Microsoft's 2006 patent application (which claims this approach was a new invention), the Petition claims that it would have been obvious to try to combine elements of file-based solutions with sector-based solutions in 2000.

xi. Petitioner's alleged motivations to combine are deficient.

276. Petitioners' Expert's Declaration at ¶ 50 claims that the combination [of Kenley and Long] "serves the goal of "reduc[ing] the time associated with

conventional data migration schemes" taught by Kenley, which is consistent with the goal of providing an "efficient, reliable and cost[-]effective architecture for disk fault tolerance" taught by Long." Petitioners' Expert's Declaration (*Id.*) also claims that Long provides "creating backup data after each modification, thus avoiding the potential loss of data created in between the timed backups of Kenley and other incremental backup systems."

277. Petitioners' Expert's Declaration at ¶ 53 also claims that "a POSITA reading Long, with its discussion of applications modifying data, would immediately contemplate use of the Long approach within file-based systems, for preserving data stored in files within that system."

278. However, Petitioners' own reference, Schneider-PCT, explicitly contradicts the above claims and teaches away from "creating backup data after each modification" at the file level due to the immediate potential of overflowing the disk drive (overwhelming Long's circular buffer approach).

> "**Keeping historical copies of files as they progress in time has the drawback** of eventually forcing the user to manage the archives and purge copies in order to avoid overflowing the disk. Obviously, **one cannot keep a backup of all files on a disk whenever they are changed** for all of time without **requiring an unlimited disk**, which does not exist." EX1020, 6:18-22; emphasis

279. The above claims of Petitioners' Expert's Declaration also ignore the difficulties and challenges of the alleged combination of Long and Kenley. As

described earlier in the section titled *A POSITA would not have had a reasonable expectation of success in combining the references*, sector-based solutions and file-based solutions are different approaches are different and mutually exclusive, a condition persisting until today.

280. Additionally, as described earlier in the section titled *Petitioner's patent application (EX2016) describes sector-level implementations and sector-level implementations as different and incompatible*, the above claims of Petitioners' Expert's Declaration are contradicted by Petitioner Microsoft's patent application (EX2016), which highlights the inability to capture and restore files in sector-level implementations.

> "[0004] Embodiments of the invention **capture and restore one or more logical volumes** of data on computer readable media. In an embodiment, the invention **images the volumes while the volumes are offline to include files associated with an operating system stored on the volumes**. Aspects of the invention further **capture volume and partition information regarding the physical layout of the volumes** such that the volumes may be restored to a state that is logically equivalent to a prior state. Further, the system supports **capture of selected data**. The system also enables efficient and effective defragmentation of the volumes." EX2016, 0004; emphasis added

281. Petitioners' Expert's Declaration at ¶ 53 claims that Kenley explains ""stor[ing] data organized as files" and using file attributes to track the backup location of files enables the data to be more easily and transparently recovered from archive storage media for use by an application."

282. However, file attributes are available only at the file level, and Long, which is sector based, has no ability to access, read, track, save, or restore.

283. Petitioners' Expert's Declaration at ¶ 54 claims that Long and Kenley "are complementary, as each teaches at least three tiers of storage—primary storage (e.g., RAM or semiconductor memory buffer), secondary storage (e.g., hard disk), and archive or backing storage (e.g., removable or external storage devices)—for use in working with and backing up data. ... copying data to a semiconductor memory buffer and then writing the data to audit buffer on disk ... moving the data from the audit buffer on disk to archive media."

284. However, such argument is unavailing, because computer systems have had these three tiers, primary storage, secondary storage, and backing storage, since at least the 1960s. In my computer engineering job at Emirtel starting between 1978 and 1980, I used computers that had these three tiers of data storage, and I programmed software that copied data from primary storage (RAM) to secondary storage (disk drive) then to backing storage (tape drive).

285. Petitioners' Expert's Declaration at ¶ 54 also claims that Kenley "teaches "'transparent' access to digital data" stored in archive storage media, with "access that entails no significant delay in ongoing processes" ... This "transparent" archiving aligns with Long's teachings on a system that "allows the

archiving to take place in the background so that the user may continue processing during an archive operation."

286. First, there is nothing unique about Kenley's and Long's transparent backup. Other than manual backup, the other two most commonly used backup approaches are scheduled and mirroring, both of which are typically transparent. In other words, the transparency of Kenley and Long is not a motivation to combine.

287. As discussed previously, file systems are implemented at a higher level than the hardware-specific device drivers and that file systems are generally hardware agnostic. Kenley's file-level transparency is inapplicable to Long, and Long's sector-level transparency is inapplicable to Kenley.

288. Petitioners' Expert's Declaration at ¶ 55 claims that "Kenley's teaching of a database-held system catalog that "stor[es] and provid[es] transparent access to catalogued file attribute data" ... dovetails with Long's teachings on an audit header for similarly maintaining data on the current location of backup copies of data.... Kenley's system catalog "enhances transparent access to file attribute data, independent of whether the file is stored in the secondary storage element or has been previously migrated from the secondary storage element" to the backing store. ... Thus, a POSITA would naturally augment the catalog database of Kenley

to store relevant information from the audit header taught by Long, thus "enhanc[ing] transparent access.""

289. However, Kenley's system catalog with file attribute data is unusable in Long's sector-based solution, because Long, operating at a lower level than Kenley, does not and cannot access file attribute data. Similarly, Long's sector-based audit header information is unusable in Kenley's file-based solution, because Kenley, operating at a higher level than Long, does not deal with sector data.

290. Petitioners' Expert's Declaration at ¶ 56 claims that "Long teaches a straightforward modification-driven archival approach that applied to known multi-tier backup–archival systems. ... A POSITA would have easily applied that approach in a file-based system such as Kenley's."

291. First, Long's modification-driven archival approach is based on detecting writes on sectors, not files, and is therefore inapplicable to Kenley, which operates at the file level.

292. Second, as described earlier in the section titled *A POSITA would not have had a reasonable expectation of success in combining the references*, sector-based solutions and file-based solutions are different approaches are different and mutually exclusive, a condition persisting until today.

293. Third, as described earlier in the section titled *Petitioner's patent application (EX2016) describes sector-level implementations and sector-level*

implementations as different and incompatible, capturing and restoring files is infeasible in sector-level implementations.

294. Petitioners' Expert's Declaration at ¶ 57 claims that "reasonable expectation of success, and overall obviousness, of implementing Long's technique for a file-based system such as Kenley's is confirmed by another prior-art patent, ... that expressly describes use of Long's audit buffer technique for saving a file's data before the file is altered." Petitioners' Expert's Declaration at ¶ 57 then cites to Schneider's Move Method and Figure 38.

295. First, as described in the section titled *Schneider-US teaches away from Long and the Move Method as slow*, Schneider-US repeatedly teaches away from Long by describing Long and the Move Method as slow, introducing delays, causing performance suffering, causing problematic overhead, and sacrificing performance.

296. Second, as described in the section titled *Schneider-US teaches that the Move Method is outside the operating system*, Schneider-US teaches that the Move Method is outside the operating system, which is consistent with the teachings of Long (sector based, not file based) unmodified. Files are handled by the operating system and its file system.

297. Third, as described in the section titled *Petitioners' Expert's Declaration's Schneider-US citations do not show Long operating at file level*,

Schneider-US does not disclose that Long is modified to make it file based or that Long is operating at file level.

298. Fourth, in citing Schneider-US's Figure 38 as evidence of using Long in a file-based implementation is flawed. Schneider's Figure 38 describes Schneider's invention, not Long's invention. For example, Schneider-US teaches handling fragmented files, which Figure 38 illustrates. Yet, Long does not disclose, suggest, or imply handling fragmented files. Indeed, Long operates at the sector level and has no access to files, let alone knowledge about fragmented files.

299. Fifth, Schneider-US's companion, Schneider-PCT, teaches away from file-level implementations.

300. Sixth, Schneider-US's related filing, Schneider-PCT (EX1020, 24:20-24), warns that file-level implementation is difficult to implement, prove, and correct, which a POSITA would have understood as teach no reasonable expectation of success.

IV. GROUND 2: CLAIMS 1-6 AND 8-11 OF THE '478 PATENT AND CLAIMS 1-6, 8-20, AND 23-31 OF THE '348 PATENT ARE NOT OBVIOUS OVER CABRERA, SCHNEIDER-PCT, AND WHITING.

301. In Ground 2, Petitioners' Expert's Declaration overlooks pivotal differences between the teachings of the '478 and '348 Patents on one hand versus the disclosures of Cabrera, Schneider-PCT, and Whiting, individually and in combination, on the other hand.

302. Petitioners' Expert's Declaration neglects to address critical deficiencies in Cabrera, Schneider-PCT, and Whiting, and in the combination of Cabrera, Schneider-PCT, and Whiting pertaining to the teachings of the '478 and '348 Patents.

303. I describe these differences and deficiencies below.

A. Schneider-PCT

304. Petitioners' Expert's Declaration relies on Schneider-PCT for the '478 and '348 claim features of "detecting an instruction to perform an operation on an operating file," specifically for '478 claim elements 1.2, 8.2, 9.2, 10.2, 11.2, 1.3, 8.3, 9.3, 10.3, and 11.3 and for '348 claim elements 1.2, 8.2, 12.2, 13.2, 14.2, 15.2, 1.3, 8.3, 12.3, 13.3, 14.3, and 15.3. 78 EX1003, ¶¶ 138-139, 141; 79 EX1003 ¶¶ 159-160, 162. A POSITA would have understood that Schneider-PCT does not teach these features for several reasons.

305. In fact, a POSITA would have understood that Schneider-PCT suffers from multiple deficiencies. Before explaining the deficiencies, main Schneider-PCT features used in Petitioners' Expert's Declaration are discussed.

a. Schneider-PCT's Main Feature Used in Petitioners' Expert's Declaration

306. Petitioners' Expert's Declaration relies on Schneider-PCT mainly for allegedly teaching "detecting an instruction by an operating system to perform an operation on an operating file." 78 EX1003, ¶¶ 138-139; 78 EX1003, ¶¶ 159-160

307. A POSITA would not have relied on or even considered Schneider-PCT, because Schneider-PCT teaches away from file-level implementations and because Schneider-PCT discloses sector-based write interception and **not** file-based write interception.

b. Schneider-PCT Deficiency #1: Schneider-PCT teaches away from any file-level implementations.

308. A POSITA would not rely on Schneider-PCT, because it teaches away from file-level implementation and, instead, recommends sector-level implementation.

i. Schneider-PCT teaches away from keeping backups of files as they change.

309. Schneider-PCT teaches away from keeping backup copies of files while the files are being changed, which the '478 and '348 Patents recite.

> "**Keeping historical copies of files as they progress in time has the drawback** of eventually forcing the user to manage the archives and purge copies in order to avoid overflowing the disk. Obviously, **one cannot keep a backup of all files on a disk whenever they are changed** for all of time without requiring an unlimited disk, which does not exist." EX1020, 6:18-22; emphasis

310. From these teachings, A POSITA would have understood that Schneider-PCT recognizes that file-level implementations and sector-level implementations are fundamentally different, not complimentary, and mutually exclusive alternatives. A POSITA would have also understood that attempting to

apply techniques from one to the other results in undesirable outcomes. Additionally, for at least these reasons, a POSITA would have been discouraged from combining any file-level implementation from a sector-level implementation and not led in such a direction.

ii. Schneider-PCT teaches that file-level implementation is difficult to implement.

311. Schneider-PCT teaches that file-level implementation is difficult to implement, prove, and correct.

> "**Although the present invention at the file level is possible it may be more difficult to implement, prove correct, and isolate as a subsystem**. Such an implementation could save an entire prior file state just before a file is opened and the intention to modify is clear, or it could attempt to only save prior states of the portion of the file being modified." EX1020, 24:20-24; emphasis

312. From these teachings, A POSITA would have understood that Schneider-PCT recognizes that file-level implementations and sector-level implementations are fundamentally different, not complimentary, and mutually exclusive alternatives. A POSITA would have also understood that attempting to apply techniques from one to the other results in undesirable outcomes. Additionally, for at least these reasons, a POSITA would have been discouraged from combining any file-level implementation from a sector-level implementation and not led in such a direction.

iii. Schneider-PCT teaches that sector-level implementations and file-level implementations are different and require different approaches, but sector-level implementations are faster.

313. Schneider-PCT teaches that sector-level implementations and file-level implementations are distinct and fundamentally different, requiring different approaches, and also teaches that sector-level implementations are a better and faster approach.

314. For example, Schneider-PCT teaches that a sector-based implementation provides stable points in time to revert to, which a file-based implementation does not.

> "An underlying assumption in the **sector (versus file) based implementation of the present invention** is that the disk will often be left in stable states, thus providing points in time to which one can revert and recover information." EX1020, 19:9-11

315. Schneider-PCT teaches that with sector-level backup, disk drives can be read sequentially from start to end, which is faster than seeking around the disk to read individual files in the case of file-level backup, since files may be fragmented; i.e., the sectors of individual files may not be stored sequentially on the disk.

> "**Reading the disk sequentially from start to end is much faster than jumping around on the disk** reading each file one at a time. This is because often a file is not stored continuously in one area of the disk, but may be spread out and intermixed

> with other files across the entire disk." EX1020, 2:25-28; emphasis added

316. Schneider-PCT teaches that with sector-level restoration using disk images, the entire disk is overwritten with the archived data, whereas with file-level restoration, individual files may be restored, which is preferred if only certain files are to be restored.

> "it is possible to efficiently restore one or more files to a disk that may be both different and already containing data (i.e., when restoring a saved disk image all prior data on a disk is overwritten)." *Id.*, 2:29-3:2

317. From these teachings, a POSITA would have understood that Schneider-PCT recognizes that file-level implementations and sector-level implementations are fundamentally different, not complimentary, and mutually exclusive alternatives. A POSITA would have also understood that attempting to apply techniques from one to the other results in undesirable outcomes. Additionally, for at least these reasons, a POSITA would have been discouraged from combining any file-level implementation and a sector-level implementation and not led in such a direction.

iv. Schneider-PCT teaches that sector-level archival provides better protection than file-level archival.

318. According to Schneider-PCT, sector-level archival protects data belonging to files as well as data not belonging to files (e.g., non-file based

operating system data), therefore providing fault tolerance protection and resilience that are unavailable with file-level protection.

> "By **implementing the <u>disk sector embodiment</u> of the present invention below the operating system (as a pre-disk controller)** the present invention is de-coupled from the operating system. Since this embodiment of the present invention can revert a disk back to an earlier state it **can recover from bugs in the operating system** that might otherwise cause catastrophic information loss by a single improper disk write. Backup techniques that are tightly coupled with an operating system and its filing system are less able to recover from bugs in themselves." EX1020, 10:28-11:4; emphasis added

> "There are **two down sides to having the mapping as part of the operating system** proper. First, in the event of an **unexpected crash** the maps may not get fully written out and thus the disk could be left in an extremely confused state. … The second reason why it is not desirable to incorporate the saving of original states in the operating system is **complexity and bugs**. Intertwining an operating system's filing system with the old state mapping **leads to bugs that can corrupt everything**. However, by separating out the management of the original old disk states from the operating system, this relatively simple management system could recover from the operating system's filing system bugs." *Id.*, 13:14-17 and 14:4-9; emphasis added

319. From these teachings, A POSITA would have understood that Schneider-PCT recognizes that file-level implementations and sector-level implementations are fundamentally different, not complimentary, and mutually exclusive alternatives. A POSITA would have also understood that attempting to apply techniques from one to the other results in undesirable outcomes.

Additionally, for at least these reasons, a POSITA would have been discouraged from combining any file-level implementation and a sector-level implementation and not led in such a direction.

v. Schneider-PCT teaches that sector-level archival is operating system independent.

320. According to Schneider-PCT, sector-level archival is **operating system independent**, making sector-level implementation more versatile and more resilient than file-level archival.

> "By **implementing the disk sector embodiment** of the present invention below the operating system (as a pre-disk controller) the present invention is **de-coupled from the operating system**. Since this embodiment of the present invention **can revert a disk back to an earlier state it can recover from bugs in the operating system that might otherwise cause catastrophic information loss** by a single improper disk write. **Backup techniques that are tightly coupled with an operating system and its filing system are less able to recover from bugs in themselves**." EX1020, 10:28-11:4; emphasis added

321. For at least these reasons, a POSITA would have understood that sector-level implementation is preferred, since it is de-coupled from the operating system and operating system independent, and, therefore, does not teach "detecting an instruction by an operating system."

322. A POSITA would have been discouraged from combining any file-level implementation and a sector-level implementation as such direction is "less able to recover from bugs" and often leads to a catastrophic total loss of data.

323. Additionally, from these teachings, A POSITA would have understood that Schneider-PCT recognizes that file-level implementations and sector-level implementations are fundamentally different, not complimentary, and mutually exclusive alternatives. A POSITA would have also understood that attempting to apply techniques from one to the other results in undesirable outcomes.

vi. Schneider-PCT teaches away from managing the storage required for backup by file and states that management by time is advantageous.

324. A POSITA would have understood that Schneider-PCT teaches away from managing the storage required for backup by file and recommends managing storage by time instead.

> "it is **advantageous to manage the storage required for backup <u>not by file, but by time</u>**. Using the technology of the present, the invention, information is maintained for a reasonable period of time and then is automatically discarded. What is included in the backup information is significantly all of the activity to the disk. **This allows a user to return to any disk state at any time** up to a limit. This limit is determined by the amount of available backup storage and the rate to which information is written (user activity)." EX1020, 10:10-16; emphasis added

325. From these teachings, A POSITA would have understood that Schneider-PCT recognizes that file-level implementations and sector-level implementations are fundamentally different, not complimentary, and mutually exclusive alternatives. A POSITA would have also understood that attempting to

apply techniques from one to the other results in undesirable outcomes. Additionally, for at least these reasons, a POSITA would have understood that sector-level implementation is preferred, since it allows a user to return to any disk state at any time and would not have been led in a direction of backing up a file approach over a time approach.

vii. Schneider-PCT teaches away from mapping and redirecting writes into the operating system.

326. Schneider-PCT teaches away from mapping and redirecting writes into the operating system.

> "There are **two down sides to having the mapping as part of the operating system** proper. First, in the event of an **unexpected crash** the maps may not get fully written out and thus **the disk could be left in an extremely confused state**." EX1020, 13:14-17; emphasis added
>
> "The second reason why it is not desirable to incorporate the saving of original states in the operating system is **complexity and bugs**. **Intertwining an operating system's filing system with the old state mapping leads to bugs that can corrupt everything**. However, by separating out the management of the original old disk states from the operating system, this relatively simple management system could recover from the operating system's filing system bugs." *Id.*, 14:4-9; emphasis added

327. From these teachings, A POSITA would have understood that Schneider-PCT recognizes that file-level implementations and sector-level implementations are fundamentally different, not complimentary, and mutually exclusive alternatives. A POSITA would have also understood that attempting to

apply techniques from one to the other results in undesirable outcomes. Additionally, for at least these reasons, a POSITA would have understood that sector-level implementation is preferred, since it allows a user to return to any disk state at any time and not led in a direction of backing up a file over time.

viii. Summary of Schneider-PCT teach-aways from any file-level implementations

328. First, Schneider-PCT teaches away from keeping backups of files as the files change.

329. Second, Schneider-PCT teaches that file-level implementation is difficult to implement and often leads to a catastrophic loss of data.

330. Third, Schneider-PCT teaches that sector-level implementations and file-level implementations are fundamentally different and require different approaches, but sector-level implementations are faster.

331. Fourth, Schneider-PCT teaches that sector-level archival provides better protection than file-level archival.

332. Fifth, Schneider-PCT teaches that sector-level archival is decoupled from the operating system and operating system independent.

333. Sixth, Schneider-PCT teaches away from managing the storage required for backup by file and states that management by time is advantageous.

334. Seventh, Schneider-PCT teaches away from mapping and redirecting writes into the operating system.

c. Schneider-PCT Deficiency #2: Schneider-PCT does not disclose "detecting an instruction by an operating system to perform an operation on an operating file" or "the step of detecting an instruction by a resident program in a computing device for performing an operation on an operating file."

335. For at least the reasons explained below, a POSITA would have understood that Schneider-PCT does not disclose "detecting an instruction by an operating system to perform an operation on an operating file" as recited in the independent claims of the '478 Patent and "detecting an instruction by a resident program in a computing device for performing an operation on an operating file" as recited in the independent claims of the '348 Patent. Throughout this section and its subsections, I will refer to both limitations as "detecting an instruction by an operating system [or resident program] to perform an operation on an operating file."

336. Instead, at best, Schneider-PCT only discloses sector-based write operation interception.

i. Schneider-PCT discloses sector-based write operation interception, not file-based write operation interception.

337. Tellingly, Petitioners' Expert's Declaration does not provide a single citation showing interception of write operations on files in Schneider-PCT, because Schneider-PCT discloses interception of sector-based write operations **only** and not interception of file-based write operations.

338. A POSITA would have realized that Schneider-PCT's interception of sector-based write operations (but not file-based write operations) was consistent Schneider-PCT's teachings away from file-based implementations.

339. Additionally, a POSITA would have understood that Schneider-PCT does not disclose "detecting an instruction by an operating system to perform an operation on an operating file" of the '478 and '348 Patents, which are file based, not sector based.

340. Schneider-PCT mentions write operation interception / trapping multiple times. I address each separately, below.

1. First Schneider-PCT Disclosure of Write Operation Interception: EX1020, 11:30-12:8

341. The first Schneider-PCT Disclosure of Write Operation Interception is reproduced below as follows:

> "As Writes are Generated
> The history buffer works by recording the original state of **sectors** on a disk prior to being changed. The time of change is also recorded, although it is not essential, in some cases it is necessary to know the order in which changes have been made, but not the time these are made. The process is illustrated in Figure 1. Thus, if a **sector** 10 contains value A, and a request to write a value B occurs, our method involves **intercepting the write**, reading the sector location 10 (picking up the A value), writing this original value into the circular history buffer, and then returning to complete the write of the new B value." EX1020, 11:30-12:8; emphasis added

342. The absence of the word "file" is noteworthy, and a POSITA would have understood from this disclosure as examined that Schneider-PCT is directed towards intercepting sector write activities and **not** intercepting file write activities.

343. A POSITA would have understood from this disclosure that Schneider-PCT does not teach intercepting file write activities and would have understood "intercepting the write" as sector write operations, not file write operations.

344. Schneider-PCT's Figure 1, which is referenced in the above excerpt, is reproduced below.

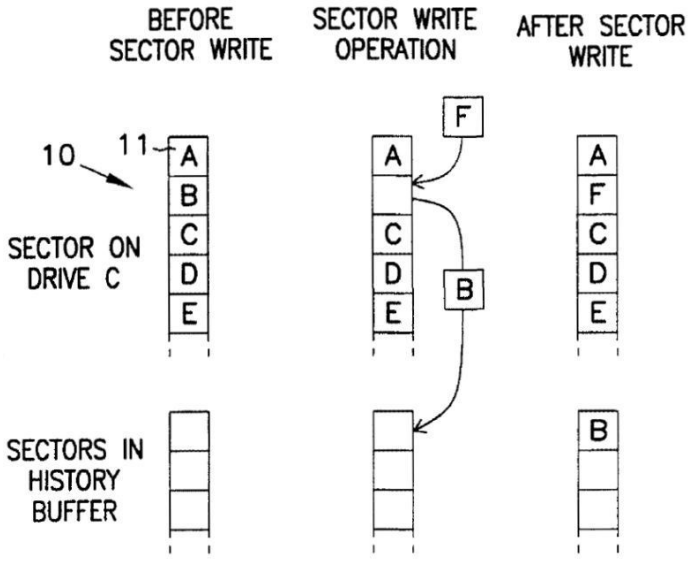


FIG. 1

345. The figure includes descriptive labels at the top of each section: "BEFORE **SECTOR** WRITE," "**SECTOR** WRITE OPERATION," and "AFTER **SECTOR** WRITE." (Emphasis added)

346. FIG. 1 also includes descriptive labels on the right of each section: "**SECTOR** ON DRIVE C" and "**SECTORS** IN HISTORY BUFFER[.]" (Emphasis added)

347. A POSITA would have understood that Schneider-PCT discloses at best sector-based write operation interception but not a file write interception.

2. Second Schneider-PCT Disclosure of Write Operation Interception: EX1020, 30:4-11

348. The second Schneider-PCT Disclosure of Write Operation Interception is reproduced below.

> "How to Intercept Disk Activity
> The technology of the present invention is based on intercepting and modifying transfers to and from a disk. The method of doing such is operating system dependent, but typically involves writing a device driver for a hard disk that is inserted in front of that normally used. In the case of DOS the techniques are well known to accomplish this effect. See the INT 13 request. As already mentioned, a partial hardware implementation could involve putting logic of the present invention right in the disk controller." EX1020, 30:4-11

349. Petitioners' Expert's Declaration relies on this Schneider-PCT paragraph (EX1020, 30:4-11) for allegedly teaching "detecting an instruction to

perform an operation on an operating file" as recited in various claim elements. 78 EX1003 ¶¶ 134, 138; 79 EX1003 ¶¶ 155, 159

350. A POSITA would not have had this understanding and, instead, would have understood that Schneider-PCT does not teach "detecting an instruction to perform an operation on an operating file" for at least the following reasons.

351. "File" does not appear anywhere in the paragraph. Each part of the foregoing paragraph will be separately reviewed to further demonstrate that Schneider-PCT discloses intercepting sector write activities and not file activities.

2-1. Schneider-PCT EX1020, 30:4 of 30:4-11

352. The first part of the foregoing Schneider-PCT paragraph discloses "How to Intercept Disk Activity." EX1020, 30:4; emphasis added

353. As shown, the paragraph is directed towards intercepting disk activity, and a POSITA would have understood that disk activity is directed towards intercepting sector-level activities and not directed towards file-level activities as the term "disk activity" is not equivalent to "file activity." Therefore, a POSITA would have understood that the Schneider-PCT paragraph does not teach "detecting an instruction to perform an operation on an operating file."

2-2. Schneider-PCT EX1020, 30:5-6 of 30:4-11

354. The second section of the foregoing Schneider-PCT paragraph of discloses "The technology of the present invention is based on intercepting and modifying transfers to and from a disk." EX1020, 30:5-6; emphasis added

355. A POSITA would have understood that intercepting and modifying transfers to and from a disk are directed towards sector-level transfers and not file level transfers. The term "transfers to and from a disk" does not mean "transfers to and from a file" or "file transfers to and from a disk." Therefore, a POSITA would have understood that the Schneider-PCT paragraph does not teach "detecting an instruction to perform an operation on an operating file."

2-3. Schneider-PCT EX1020, 30:6-8 of 30:4-11

356. The third section of the foregoing Schneider-PCT paragraph of discloses "The method of doing such is operating system dependent, but typically involves writing a device driver for a hard disk that is inserted in front of that normally used." EX1020, 30:6-8; emphasis added

357. A POSITA would have understood that the foregoing does not disclose intercepting whether file based or otherwise, since there is simply no mention of files. Also, a "device driver for a hard disk" does not indicate handling files or that the device driver is file based. Instead, a POSITA would have understood that the "operating system dependent" indicates that the interface

between the operating system and device drivers is operating system dependent. In addition, a POSITA would have understood that a resident program, as taught in the ’478 and ’348 Patents, is not discussed. Therefore, a POSITA would have understood that the Schneider-PCT paragraph does not teach “detecting an instruction to perform an operation on an operating file.”

2-4. Schneider-PCT EX1020, 30:8-9 of 30:4-11

358. The fourth section of the foregoing Schneider-PCT paragraph of discloses “<u>In the case of DOS the techniques are well known to accomplish this effect. See the INT 13 request</u>.” EX1020, 30:8-9; emphasis added

359. As described in the following section, Interrupt 13h (AKA INT 13h or INT 13) is a BIOS (or device driver) service that offers disk drive functions that are only sector-based (or block-based) and does not offer any functions that are file-based.

360. Therefore, a POSITA would have understood that this paragraph teaches a sector-based interception and not file-based interception.

361. A POSITA would have also understood that Schneider-PCT does not teach “detecting an instruction to perform an operation on an operating file.”

2-5. What Is Interrupt 13h (INT 13h)?

362. A POSITA would have understood that Interrupt 13h (AKA INT 13h or INT 13) is a BIOS service that provides disk drive services, specifically sector-

based and block-based functions but not file-based functions. BIOS provides the operating system and application programs with services for interfacing with hardware and devices (including disk drives). See also the earlier section of this Declaration titled *BIOS*.

363. The BIOS hardware interface services are called through software interrupts. In other words, the operating system or an application program invokes or calls BIOS services through software interrupts. EX2006

364. BIOS services are assigned various unique interrupt numbers (commonly referred to as hexadecimal numbers), for example, 10h for video, 14h for serial, 16h for keyboard, and 17h for parallel printer. EX2006, p. 80. Exhibit EX2006 p. 80 and Exhibit EX2009 pp. 7-8 show the BIOS interrupts and their corresponding services. As can be seen from the list, there are no file-based functions or interrupts.

365. The interrupt dealing with disk drives and floppy drives is interrupt 13h (INT 13h). EX2006, p. 80

366. BIOS interrupt services may require parameters. For example, for the video interrupt (INT 10h), one such parameter is video mode (e.g., color, black and white, or monochrome). Interrupt service parameters are passed from the operating system or application programs to the BIOS or device driver through CPU registers.

367. The disk drive interrupt, INT 13h, provides various disk drive services, which are passed through CPU registers.

368. Exhibit EX2006 pp. 43-51 and Exhibit EX2009 pp. 205-211 show the various INT 13h disk drive parameters and their corresponding functions, such as read sectors (CPU register AH is set to 02h) and write sectors (CPU register AH is set to 03h). INT 13h services are all sector (or block) based. There are no file-based services.

369. Below is a table that summarizes all the INT 13h functions. As can be seen from the table, **there are no file commands**.

INT 13h Parameter	Function
AH = 00h	Reset Disk System
AH = 01h	Get Status of Last Drive Operation
AH = 02h	Read Sectors From Drive
AH = 03h	Write Sectors To Drive
AH = 04h	Verify Sectors
AH = 05h	Format Track
AH = 06h	Format Track Set Bad Sector Flags
AH = 07h	Format Drive starting at Track
AH = 08h	Read Drive Parameters
AH = 09h	Initialize Disk Controller
AH = 0Ah	Read Long Sectors From Drive
AH = 0Bh	Write Long Sectors To Drive
AH = 0Ch	Move Drive Head To Cylinder
AH = 0Dh	Reset Disk Drives
AH = 0Eh	Controller Read Test
AH = 0Fh	Controller Write Test
AH = 10h	Test Whether Drive Is Ready
AH = 11h	Recalibrate Drive
AH = 12h	Controller RAM Test
AH = 13h	Drive Test

AH = 14h	Controller Diagnostic
AH = 15h	Read Drive Type
AH = 16h	Detect Media Change
AH = 17h	Set Media Type For Format (DOS versions <= 3.1)
AH = 18h	Set Media Type For Format (DOS versions >= 3.2)
AH = 19h	Park Heads

370. In addition, disk read and write functions use cylinder-head-sector (CHS) or logical block address (LBA) locations.

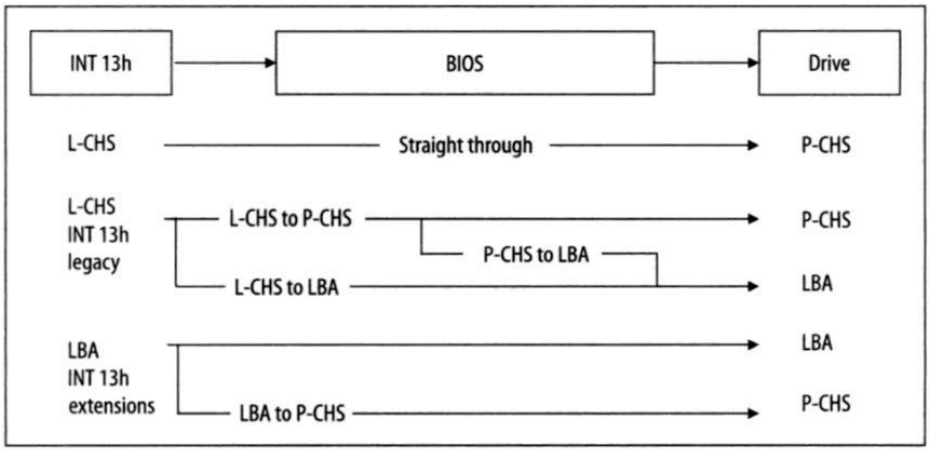


Fig. 5.20 Translation options.

EX2007, p. 132

371. INT 13h is available in DOS-based systems and in other systems. Exhibit EX2006 p. 32

372. In summary, INT 13h is a BIOS (or device driver) service that offers disk drive functions, specifically sector-based or block-based functions but not file-based functions.

2-6. Schneider-PCT EX1020, 30:9-11 of 30:4-11

373. The fifth section of the foregoing Schneider-PCT paragraph of discloses "As already mentioned, a partial hardware implementation could involve putting logic of the present invention right in the disk controller." EX1020, 30:9-11; emphasis added

374. A POSITA would have understood that this paragraph teaches a sector-based interception and not a file-based interception, because a POSITA would have understood (as described in the following section) that disk controllers deal with sector-based or block-based functions, not file-based functions.

375. A POSITA would have also understood that Schneider-PCT does not teach "detecting an instruction to perform an operation on an operating file."

2-7. What Is a Disk Controller?

376. Disk controller (AKA hard disk controller, disk drive controller) is an electronic circuit that controls the operation of a disk drive and allows a computer to communicate with the disk drive, including transferring data to and from the disk drive. The disk controller includes various functions: read-write channel circuitry to control reading and writing from and to the media (the platters), servo control circuitry to control the precise movement of heads, host interface circuitry to control the interface (e.g., SCSI, ATA/IDE/EIDE, etc.) with the host computer, buffer memory, and firmware to manage the various functions of the controller.

The disk controller is typically attached to the disk drive mechanical assembly. The disk controller and the disk drive mechanical assembly become one unit, sold as an internal disk drive or an external disk drive. In an external disk drive, the disk controller is visually identifiable, namely the circuit board on the disk drive. EX2007, p. 119; EX2010, p. 263

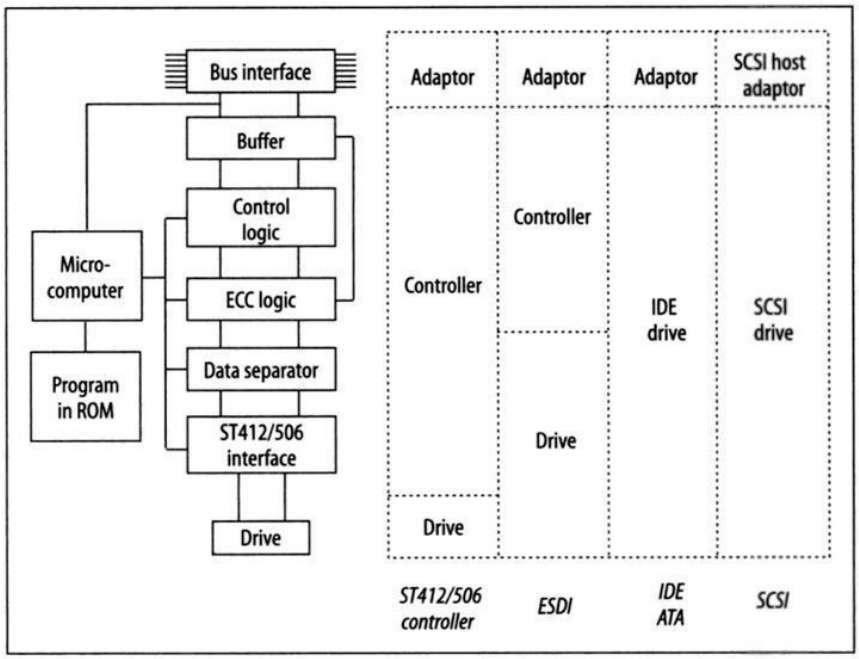


Fig. 5.15 Controllers and drives.

EX2007, p. 119

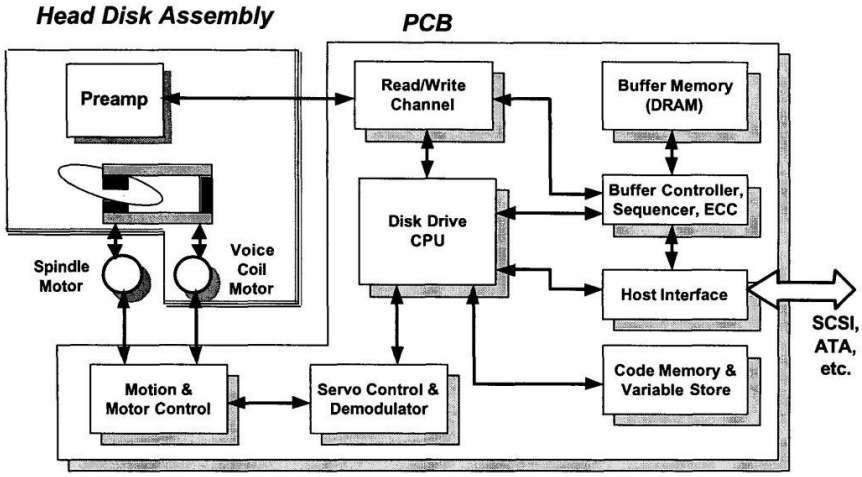


Figure 1. Hard disk drive subsystem

EX2010, p. 263

377. As of 2000, the most common data storage interfaces (AKA protocols or standards using proper data storage vernacular) were SCSI and IDE (ATA, EIDE). At the time of the invention, SCSI, IDE, and the other (less commonly used) data storage interfaces were all sector based, not file based.

378. For SCSI, for example, EX2011 was an official ANSI SCSI-2 standard. I included all the pages dealing with direct-access devices (e.g., disk drives). All the commands that require device locations (e.g., read and write) rely on logical block address (LBA) parameters. There are no file commands.

379. For ATA, for example, EX2012 was an official ANSI ATA draft standard. I included all the pages dealing with disk drive commands. All the commands that require locations (e.g., read and write) rely on cylinder-head-sector (CHS) or logical block address (LBA) parameters. There are no file commands.

380. Disk controllers are related to host bus adapters.

381. A host bus adapter (HBA, AKA host adapter) is a circuit board that resides in the computer (host computer) and connects the computer with the disk controller through SCSI, IDE, or other interface.

HBA Host Bus Adapter. A printed circuit board that acts as an interface between the host microprocessor and the disk controller. The HBA relieves the host microprocessor of data storage and retrieval tasks, usually increasing the computer's performance time. A host bus adapter (or host adapter) and its disk subsystems make up a disk channel.

EX2013, p. 398

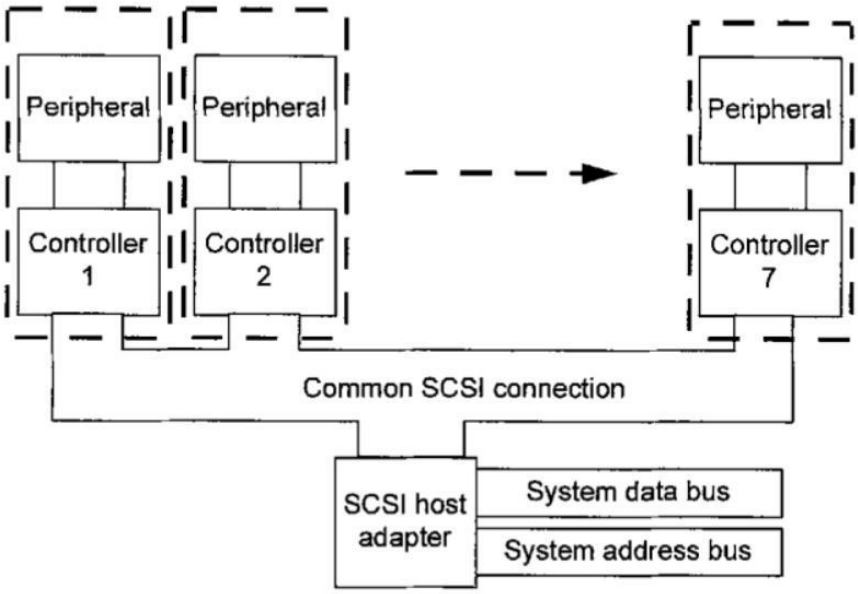


Figure 6.3 SCSI interface

EX2003, p. 143

382. The SCSI or IDE commands between the host adapter and the disk controller are sector based, not file based.

383. In summary, a POSITA would have understood that disk controllers deal with sector-based or block-based functions, but not file-based functions.

3. Third Schneider-PCT Disclosure of Write Operation Interception: EX1020, 23:3-8

384. The third Schneider-PCT Disclosure of Write Operation Interception is reproduced below.

> "As a point of the present invention's **<u>sector</u> implementation**, the simulated drive D is "created" by **trapping all read and write disk transfers**. For any read accesses a map is consulted that indicates the true **location of desired disk <u>sectors</u>**." EX1020, 23:3-8; emphasis added

385. A POSITA would have understood that this disclosure is directed towards a sector-based interception and not a file-based interception. Moreover, this disclosure does not even mention of files and does not disclose "detecting an instruction to perform an operation on an operating file."

4. Fourth Schneider-PCT Disclosure of Write Operation Interception: EX1020, 26:7-8

386. The fourth Schneider-PCT Disclosure of Write Operation Interception is reproduced below.

> "As Reads are Generated
> In the case where a **disk is entirely reverted back** in time the associated algorithm can be pretty straight forward: walk through the history buffer moving original states back in place until the desired time is reached. Note that during this copy, the **writes are still trapped** and old states recorded so that this reversion is in fact reversible." EX1020, 22:17-22; emphasis added

387. A POSITA would have understood that this disclosure is directed towards a sector-based interception of writes and not a file-based interception. Moreover, this disclosure does not even mention of files and does not disclose "detecting an instruction to perform an operation on an operating file."

5. Fifth Schneider-PCT Disclosure of Write Operation Interception: EX1020, 26:7-8

388. The fifth Schneider-PCT Disclosure of Write Operation Interception is reproduced below.

> "The technology of the present invention involves **logging disk <u>sector</u> writes** and other activity to a circular history buffer." EX1020, 26:7-8; emphasis added

389. A POSITA would have understood that this disclosure is directed towards a sector-based interception and not a file-based interception. Moreover, this disclosure does not even mention files and does not disclose "detecting an instruction to perform an operation on an operating file."

6. In summary, Schneider-PCT discloses sector-based write operation interception, not file-based write operation interception.

390. In summary, as shown in all the citations in Schneider-PCT related to interception, Schneider-PCT discloses sector-based write operation interception, not file-based write operation interception.

391. A POSITA would have realized that Schneider-PCT sector-based write operation interception utilizes BIOS or low-level disk driver drivers for the interception rather than the operating system / file system. A POSITA would have also realized that Schneider-PCT sector-based write operation interception has no ability to intercept files, which are handled at the operating system / file system level or a resident program (as resident program is taught in the ’478 and ’348 Patents).

ii. Schneider-PCT also references other patents each of which discloses sector-based write operation interception but does not disclose file-based write operation interception.

392. Schneider-PCT further teaches sector-based write operation interception by citing other patents, each of which teaches sector-based write operation interception and does not teach a file-based write operation interception or “detecting an instruction to perform an operation on an operating file.”

393. The first patent reference disclosing sector-based write operation interception that Schneider-PCT cites is the Long patent, the same Long patent of Ground 1 (Schneider-PCT erroneously refers to Long as ’5**7**9, instead of ’5**1**9).

> “Patent No. 5,325,579, entitled "Fault Tolerant Computer with Archival Rollback Capabilities", to Long et al. ("'579 patent"). The **'579** patent discloses a storage device which includes **processing circuitry for detecting access requests to alter data in respective locations of a storage device**, and, prior to executing such requests, storing the data in such locations in an

> audit partition region of the storage device." EX1020, 6:23-29; emphasis added

394. A POSITA would have understood from the above teachings that Long is sector based and does not disclose "detecting an instruction to perform an operation on an operating file."

395. The second patent disclosing sector-based write operation interception that Schneider-PCT cites is Bemis. It appears that the patent number cited by Schneider-PCT is incorrect. The correct number should be US Patent No. 5,487,160, to Edward D. Bemis, titled "Concurrent image backup for disk storage system."

> "The Bemis patent, U.S. Patent No. 5,553,160, teaches a related method where during a backup, a **write request to a disk is trapped**. If the disk location being written has not yet been backed up, the original contents are copied to an alternate temporary storage device in order to allow the backup to proceed." EX1020, 17:17-20; emphasis added

396. A POSITA would have understood that Bemis is sector based and does not disclose "detecting an instruction to perform an operation on an operating file."

iii. A POSITA would not have relied on Schneider-PCT to modify Cabrera, since Schneider-PCT teaches away from any file-based implementations.

397. As described earlier in the section titled *Schneider-PCT Deficiency #1: Schneider-PCT teaches away from any file-level implementations*, Schneider-PCT's teach-away from file-based implementations.

398. Considering Schneider-PCT's aversion from and teach-aways against file-based implementations, it would be unlikely for Schneider-PCT to rely on or use file-based write operation interception. Indeed, Schneider-PCT uses sector-based write operation interception, not file-based write operation interception.

d. A POSITA would not have been motivated to modify Cabrera with Schneider-PCT's interception from sector based to file based.

399. A POSITA would not have been motivated to modify Schneider-PCT's interception from sector based to file based for the following reasons.

First

400. Schneider-PCT teaches away from file-level implementations, as described earlier in the section titled *Schneider-PCT Deficiency #1: Schneider-PCT teaches away from any file-level implementations*.

Second

401. Schneider-PCT teaches that sector-level archival provides better protection than file-level archival, as described earlier in the section titled *Schneider-PCT teaches that sector-level archival provides better protection than file-level archival*.

Third

402. Schneider-PCT teaches that sector-level archival is operating system independent, as described earlier in the section titled *Schneider-PCT teaches that sector-level archival is operating system independent*. Schneider-PCT also does not teach a resident program as taught in the ’478 and ’348 Patents.

Fourth

403. Schneider-PCT teaches away from managing the storage required for backup by file, as described earlier in the section titled *Schneider-PCT teaches away from managing the storage required for backup by file and states that management by time is advantageous*.

Fifth

404. Schneider-PCT teaches away from mapping and redirecting writes into the operating system or a resident program (as resident program is taught in the ’478 and ’348 Patents), as described earlier in the section titled *Schneider-PCT teaches away from mapping and redirecting writes into the operating system*.

Schneider-PCT also does not teach a resident program as taught in the ’478 and ’348 Patents.

e. A POSITA would not have considered and would not have been motivated to use Schneider-PCT to reproduce the ’478 and ’348 Patents.

405. Because of the Schneider-PCT deficiencies explained earlier, a POSITA would not have considered Schneider-PCT to reproduce the ’478 and ’348 Patents.

B. Cabrera

406. Petitioners’ Expert’s Declaration (78 EX1003 ¶¶ 120-150; 79 EX1003 ¶¶ 140-171) (1) relies on Schneider-PCT mainly for the interception claim limitation (detecting an instruction to perform an operation on an operating file), (2) relies on Whiting for the database limitation (updating the database to indicate that the archive file is located …), and (3) relies on Cabrera for the other claim limitations.

407. As described below, Cabrera suffers from multiple deficiencies. In the following sections, I describe these deficiencies.

a. Cabrera Deficiency #1: Cabrera does not disclose “detecting an instruction to perform an operation on an operating file.”

408. Cabrera does not disclose “detecting an instruction to perform an operation on an operating file.”

i. Petitioners' Expert's Declaration does not identify "an operation on an operating file," "an instruction to perform an operation on an operating file," or "detecting an instruction to perform an operation on an operating file" in Cabrera.

409. Petitioners' Expert's Declaration fails to identify "an operation on an operating file," "an instruction to perform an operation on an operating file," or "detecting an instruction to perform an operation on an operating file" in Cabrera.

410. Petitioners' Expert's Declaration (78 EX1003, ¶ 137; 79 EX1003, ¶ 158) states that "Cabrera discloses that "[a]rchive systems … typically attempt to maintain a complete history of the changes made to a particular entity, such as a particular file or storage device."

411. First, the statement does not identify "an operation on an operating file," "an instruction to perform an operation on an operating file," or "detecting an instruction to perform an operation on an operating file."

412. Second, a POSITA would have understood that maintaining a history of the changes made to a file indicates that such history is created after the supposed changes have **<u>already</u>** been made to the file. Otherwise, the history would be subject to corruption if, for example, the system crashes before a change is made to the file, rendering the last history entry invalid.

413. Third, a POSITA would have understood that maintaining a complete history of the changes made to a file does not disclose "an operation on an

operating file," let alone "an instruction to perform an operation on an operating file," let alone "detecting an instruction to perform an operation on an operating file."

414. Petitioners' Expert's Declaration (*Id.*) also states that "Cabrera further teaches that "[o]ne approach to allowing access to files while archive or backup copies are created is to duplicate the information that will be backed up or archived and 'stage' the information in a temporary storage area."

415. This statement suffers from the same flaws as the earlier statement.

416. First, the statement does not identify "an operation on an operating file," "an instruction to perform an operation on an operating file," or "detecting an instruction to perform an operation on an operating file."

417. Second, a POSITA would have understood that allowing access to files while archive or backup copies are created and duplicating the information that will be backed up or archived are performed **after** the supposed changes have been made to the files.

418. Third, a POSITA would have understood that allowing access to files while archive or backup copies are created and duplicating the information that will be backed up or archived do not disclose "an operation on an operating file," let alone "an instruction to perform an operation on an operating file," let alone "detecting an instruction to perform an operation on an operating file."

419. Petitioners' Expert's Declaration (*Id.*) then states that "Cabrera discloses that "any … triggering event" may be used to trigger the copying of data from local storage to the staging area."

420. This statement suffers from the same flaws as the two earlier statements.

421. First, the statement does not identify "an operation on an operating file," "an instruction to perform an operation on an operating file," or "detecting an instruction to perform an operation on an operating file."

422. Second, a POSITA would have reasoned that phrase "any triggering event" is overly general and not explicit and would have realized that assigning a specific meaning to the phrase is conjectural.

423. Third, a POSITA would have understood that "any triggering event" does not disclose "an operation on an operating file," let alone "an instruction to perform an operation on an operating file," let alone "detecting an instruction to perform an operation on an operating file."

424. Lastly, Petitioners' Expert's Declaration (*Id.*) states that "Cabrera also discloses a "data producer 60 [that] creates data that is to be backed up or archived to a backup or archive storage device, such as remote storage 62.""

425. This statement suffers from the same flaws as the earlier statements.

426. First, the statement does not identify "an operation on an operating file," "an instruction to perform an operation on an operating file," or "detecting an instruction to perform an operation on an operating file."

427. In fact, the statement and Cabrera citation do not even mention files.

428. Second, a POSITA would have understood that creating data that is to be backed up or archived indicates that the backup / archive is performed **after** the supposed creation of data.

429. Third, a POSITA would have understood that creating data that is to be backed up or archived does not disclose "an operation on an operating file," let alone "an instruction to perform an operation on an operating file," let alone "detecting an instruction to perform an operation on an operating file."

ii. Schneider-PCT does not cure Cabrera's deficiency of not disclosing "detecting an instruction to perform an operation on an operating file."

430. Petitioners' Expert's Declaration (78 EX1003, ¶ 137; 79 EX1003, ¶ 158) claims that "[a] POSITA would have been motivated to use detecting or intercepting "a request to write" to disk as taught by Schneider as a triggering event, because doing so would enable the archiving system to "maintain a complete history of the changes made to … a particular file."

431. However, as described earlier in the section titled *Schneider-PCT Deficiency #2: Schneider-PCT does not disclose "detecting an instruction by an*

operating system to perform an operation on an operating file" or "the step of detecting an instruction by a resident program in a computing device for performing an operation on an operating file," a POSITA would have known that Schneider-PCT does not disclose "detecting an instruction by [an operating system] [a resident program] to perform an operation on an operating file."

432. As shown in the previous section, Petitioners' Expert's Declaration fails to identify "an operation on an operating file," "an instruction to perform an operation on an operating file," or "detecting an instruction to perform an operation on an operating file" in Cabrera.

433. Therefore, a POSITA would have known that the alleged combination of Cabrera and Schneider-PCT does not disclose "detecting an instruction to perform an operation on an operating file."

iii. Cabrera's archive is performed <u>only</u> after data is already present in the local storage.

434. In Cabrera, an archive or backup operation is performed <u>only after data is present in the local storage</u>. In particular, Cabrera teaches that archive or backup only begins when one of four events occurs: (1) a defined amount of data has been stored locally, (2) a particular amount of time has elapsed, (3) command to stage data, or (4) any other triggering event.

> "if a data producer is producing data and storing it on local storage, when a **defined amount of data has been stored locally or when a particular time has elapsed**, the data may

> be copied or moved from local storage to a staging area employing sparse file technology." EX1005, 3:38-50; emphasis added
>
> "Returning now to FIG. 5, once the triggering event has occurred, whether that be the **accumulation of a given amount of data**, the expiration of a **given time delay**, the receipt of a **command to stage data**, or **any other triggering event**, execution proceeds to step 128 where data is copied from local storage to the staging file." *Id.*, 10:56-61; emphasis added

435. In all these cases, new (unarchived) data must be available in the local storage in order for the archive to proceed.

436. Cabrera explicitly states that archiving begins only **<u>after</u>** any of these four events occurs: "**once the triggering event has occurred**, …, execution proceeds to step 128 where data is copied from local storage to the staging file." *Id.*, 10:56-61; emphasis added

437. In other words, whatever event leads to initiating the backup / archive, the archive starts **after** the alleged data to be archived has been written.

438. For at least this reason alone, a POSITA would have known that the alleged combination of Cabrera and Schneider-PCT does not disclose "detecting an instruction to perform an operation on an operating file."

iv. Cabrera's "any triggering event" is not an "an operation on an operating file" and not "an instruction to perform an operation on an operating file."

439. As described above, in Cabrera, all the four events leading to initiating the backup / archive defined amount of data has been stored locally, particular amount of time has elapsed, command to stage data, or any other triggering event), occur **after** the alleged data to be archived has been written.

440. A POSITA would have understood that a triggering event cannot be "an operation on an operating file" and not "an instruction to perform an operation on an operating file," because the triggering event occurs after the data to be archived already resides in local storage.

b. Cabrera Deficiency #2: Cabrera does not disclose "creating an archive file from the operating file and storing the archive file in a temporary [first] storage location temporally proximate to the operation being performed on the operating file and responsive to detecting the instruction."

441. Cabrera does not disclose "creating an archive file from the operating file and storing the archive file in a temporary [first] storage location temporally proximate to the operation being performed on the operating file and responsive to detecting the instruction."

442. Petitioners' Expert's Declaration (78 EX1003, ¶ 141; 79 EX1003, ¶ 162) states that "Cabrera discloses copying a file 'from local storage to the staging file' in response to 'any … triggering event.'"

443. First, a POSITA would have known that Cabrera does not disclose "responsive to detecting the instruction," at least because, as described in the previous section, Cabrera does not disclose "detecting an instruction to perform an operation on an operating file."

444. Second, a POSITA would have known that Cabrera does not disclose "temporally proximate to the operation being performed on the operating file and responsive to detecting the instruction," at least because the events that initiate the backup / archive occur after the supposed data to be archived has been written and Cabrera does not stipulate any temporal restrictions on when these events in relation to the supposed writing of the data.

445. In Cabrera, an archive or backup operation is performed **only after data is present in the local storage**. In particular, Cabrera teaches that archive or backup only begins when one of four events occurs: (1) a defined amount of data has been stored locally, (2) a particular amount of time has elapsed, (3) command to stage data, or (4) any other triggering event. Cabrera does not describe or suggest any timing / temporal response limitations.

446. Petitioners' Expert's Declaration (*Id.*) then states that Cabrera's "staging file or staging area "is adapted for temporarily storing the data"" and that "Cabrera discloses that "[a]ny mechanism that performs this function [of moving data] may be utilized such as, for example, reading the data from the appropriate location and then storing a copy of the data in the staging storage area[, …] direct memory transfer and so forth.""

447. A POSITA would **not** have understood the above statements, even with the triggering event, as disclosing "creating an archive file from the operating file and storing the archive file in a temporary [first] storage location temporally proximate to the operation being performed on the operating file and responsive to detecting the instruction."

448. Petitioners' Expert's Declaration (*Id.*) also states that "it would have been obvious to use interception of a request by an application or operating system to write to disk as taught by Schneider as the triggering event to initiate the backup–archive method of Cabrera."

449. I disagree.

450. As described earlier, a POSITA would have known that Schneider-PCT does not disclose "detecting an instruction by an operating system to perform an operation on an operating file." Schneider-PCT intercepts sector writes, not file writes.

451. Petitioners' Expert's Declaration (78 EX1003, ¶ 142; 79 EX1003, ¶ 163) claims that "a POSITA would have found it obvious to create the copy of the file temporally proximate to the operation being performed "so that a logically consistent backup copy is created" because creating such a copy before overwriting a file was a well-known, fundamental technique." As support, Petitioners' Expert's Declaration offers "if a particular file or volume is to be backed up, it is usually important not to allow the contents of the file or volume to change during the backup procedure so that a logically consistent backup copy is created" from Cabrera as well as an excerpt from Long.

452. First, the above statement is contrary to Cabrera's teachings in which an archive or backup operation is performed <u>only</u> **after** <u>data is present in the local storage.</u>

453. Second, a POSITA would NOT have found it obvious to create the copy of the file temporally proximate to the operation being performed "so that a logically consistent backup copy is created" because creating such a copy before overwriting a file was a well-known, fundamental technique."

454. Third, even, assuming arguendo, if creating a copy before overwriting a file was well known and fundamental, it still does not disclose or motivate a POSITA to create a copy of the file temporally proximate to the operation being

performed. Being well known and fundamental does not relate to temporal proximity.

455. Fourth, a POSITA would have understood Cabrera's excerpt ("if a particular file or volume is to be backed up, it is usually important not to allow the contents of the file or volume to change during the backup procedure so that a logically consistent backup copy is created") as advocating backup / archive **after** the file has been updated, not before, to ensure that "a logically consistent backup copy is created."

456. Fifth, a POSITA would have understood Schneider-PCT is irrelevant, because Schneider-PCT is sector based, not file based.

457. Lastly, Petitioners' Expert's Declaration (*Id.*) argues that a POSITA would have found limitation obvious in view "copy-on-write." However, copy-on-write and the EX1007 citation proffered in Petitioners' Expert's Declaration do not stipulate temporal proximity.

458. In summary, Petitioners' Expert's Declaration fails to show that Cabrera, alone or in combination with Schneider-PCT, discloses "creating an archive file from the operating file and storing the archive file in a temporary [first] storage location temporally proximate to the operation being performed on the operating file and responsive to detecting the instruction."

459. For all these reasons, a POSITA would have known that the alleged combination of Cabrera and Schneider-PCT does not disclose "creating an archive file from the operating file and storing the archive file in a temporary [first] storage location temporally proximate to the operation being performed on the operating file and responsive to detecting the instruction."

c. Cabrera Deficiency #3: Cabrera does not teach the '478 limitation of "searching the first temporary storage location for the archive file responsive to the occurrence of a first event" or the '348 limitation of "identifying presence of the archive file in the temporary storage location responsive to the occurrence of a first event."

460. Cabrera does not teach the '478 limitation of "searching the first temporary storage location for the archive file responsive to the occurrence of a first event" or the '348 limitation of "identifying presence of the archive file in the temporary storage location responsive to the occurrence of a first event."

461. Petitioners' Expert's Declaration (78 EX1003, ¶ 144; 79 EX1003, ¶ 165) claims that "Cabrera teaches monitoring the 'amount of data in the staging area … in order to identify when a backup or archive session should be initiated.'"

462. Petitioners' Expert's Declaration (*Id.*) claims that "Cabrera further teaches an "event monitor 152" that "may be implemented in a wide variety of ways" in order "to watch for desired triggering events and to initiate appropriate action when the events occur.""

463. Additionally, Petitioners' Expert's Declaration (78 EX1003, ¶ 137; 79 EX1003, ¶ 158) previously claimed "any triggering event" as "detecting an instruction to perform an operation on an operating file." Yet, Petitioners' Expert's Declaration improperly attempts to redefine "any triggering event" to mean "first event."

464. Petitioners' Expert's Declaration (78 EX1003, ¶ 144; 79 EX1003, ¶ 165) claims that "Cabrera also teaches determining "whether more data should be appended to the staging file," where "[t]his determination may be made based on any triggering event." ... A POSITA would have understood that evaluating whether to append more data to the staging file necessarily requires searching [identifying] the staging area for the staging file, and that this searching [identifying] step would occur responsive to the triggering event."

465. A POSITA would **not** have understood determining and/or appending as requiring searching or identifying and would have not understood the above Cabrera excerpt as "searching the first temporary storage location for the archive file responsive to the occurrence of a first event" or as "identifying presence of the archive file in the temporary storage location responsive to the occurrence of a first event."

466. Additionally, as described above, Petitioners' Expert's Declaration improperly attempts to redefine "any triggering event" to mean "first event."

467. Petitioners' Expert's Declaration (78 EX1003, ¶ 144) claims that "Cabrera discloses a "means for transferring data from the staging storage area" to "remote storage communication infrastructure" which "may be implemented by any mechanism capable of retrieving data from" the staging area. ... The POSITA would have understood that searching the staging area (first temporary storage location) is a mechanism for retrieving data from the sparse file in the staging area."

468. A POSITA would **not** have understood searching the staging area as a mechanism for retrieving data and would **not** have understood the above Cabrera excerpt as "searching the first temporary storage location for the archive file responsive to the occurrence of a first event" or as "identifying presence of the archive file in the temporary storage location responsive to the occurrence of a first event."

469. In summary, Petitioners' Expert's Declaration fails to show that Cabrera, alone or in combination with Schneider-PCT, discloses "searching the first temporary storage location for the archive file responsive to the occurrence of a first event" or as "identifying presence of the archive file in the temporary storage location responsive to the occurrence of a first event."

470. For all these reasons, a POSITA would have known that the alleged combination of Cabrera and Schneider-PCT does not disclose "searching the first

temporary storage location for the archive file responsive to the occurrence of a first event" or as "identifying presence of the archive file in the temporary storage location responsive to the occurrence of a first event."

C. Whiting

471. As described below, Whiting suffers from multiple deficiencies. Before explaining the deficiencies, I describe the main Whiting features used in Petitioners' Expert's Declaration.

a. Whiting's Main Features Used in Petitioners' Expert's Declaration

472. Petitioners' Expert's Declaration (78 EX1003 ¶¶ 130, 120-150; 79 EX1003 ¶¶ 152, 140-171) relies on Whiting for the database limitation (updating the database to indicate that the archive file is located ...), specifically Whiting's "backup directory" and database.

b. Whiting Deficiency #1: Other than Whiting's main features used in Petitioners' Expert's Declaration, Whiting is deficient vis-à-vis the claim limitations of '478 and '348 Patents.

473. Other than Whiting's main features used in Petitioners' Expert's Declaration for the Whiting-Long combination, Whiting is deficient vis-à-vis the claim limitations of '478 and '348 Patents.

474. Next, I highlight two specific missing claim limitations then describe additional deficiencies.

c. Whiting Deficiency #2: Whiting does not disclose "detecting an instruction to perform an operation on an operating file."

475. Petitioners' Expert's Declaration does not rely on Whiting for this claim limitation. Nevertheless, Whiting does not disclose "detecting an instruction to perform an operation on an operating file."

d. Whiting Deficiency #3: Whiting does not disclose temporally proximate to the operation being performed on the operating file.

476. Petitioners' Expert's Declaration does not rely on Whiting for this claim limitation. Nevertheless, the temporally proximate claim limitation is irrelevant to and inapplicable in Whiting.

D. Motivation NOT to Combine Cabrera, Schneider-PCT, and Whiting

477. As described earlier in the section titled *Cabrera*, in the section titled *Schneider-PCT*, and in the section titled *Whiting*, Cabrera, Schneider-PCT, and Whiting individually suffer from multiple deficiencies.

478. Additionally, considering the deficiencies of Cabrera, Schneider-PCT, and Whiting described earlier in the section titled *Cabrera*, in the section titled *Schneider-PCT*, and in the section titled *Whiting*, and as described earlier in the section titled *A POSITA would not have considered and would not have been motivated to use Schneider-PCT to reproduce the '478 and '348 Patents*, a

POSITA would not have been motivated to use Cabrera, Schneider-PCT, or Whiting individually to reproduce the ’478 and ’348 Patents.

479. Assuming, arguendo, that a POSITA is willing to overlook Cabrera’s, Schneider-PCT’s, and Whiting’s deficiencies and to overlook the reasons for not using Cabrera, Schneider-PCT, and Whiting individually to reproduce the ’478 and ’348 Patents, a POSITA would still not be motivated to combine Cabrera, Schneider-PCT, and Whiting as contemplated in Petitioners’ Expert’s Declaration. Below, I explain the reasons.

a. A POSITA would not have been motivated to combine Cabrera, Schneider-PCT, and Whiting, because Cabrera would need intercepting file write operations at the file level, but Schneider-PCT teaches intercepting write operations at the sector level only.

480. This argument is crucial but intricate, requiring discerning analysis.

481. A POSITA would not have been motivated to combine Cabrera, Schneider-PCT, and Whiting, because Cabrera would need intercepting file write operations at the file level, but Schneider-PCT teaches intercepting write operations at the sector level only, and Whiting does not disclose any interception.

<u>What Cabrera Requires to Initiate Archives</u>

482. Cabrera’s archive is performed only if data is produced through write operations (data producers’ performing write operations); otherwise, archive is not performed.

483. Additionally, intercepting operations other than write (such as read, open, close, rename, or delete) would not cause Cabrera to perform archive or backup, because there is no unarchived data in local storage to archive.

484. Furthermore, to disclose the limitation of detecting an instruction to perform an operation on an operating file, any reference to be combined with Cabrera must disclose intercepting file writes. Intercepting other operations (e.g., open, close, delete, rename) would not suffice. Intercepting at the sector level, not file level, would not suffice.

What Schneider-PCT Does

485. However, as described earlier in the section titled *Schneider-PCT discloses sector-based write operation interception, not file-based write operation interception*, Schneider-PCT only discloses write operation interception at the sector level, not at the file level.

486. In other words, Schneider-PCT does not disclose detecting an instruction to perform a write operation on an operating file as necessary for Cabrera and as necessary for the combination Cabrera and Schneider-PCT to disclose detecting an instruction to perform an operation on an operating file. Whiting does not cure this deficiency.

487. For this reason alone, a POSITA would not have considered combining Schneider-PCT with Cabrera and would not have been motivated to

combine Cabrera, Schneider-PCT, and Whiting to implement the ’487 and ’348 Patents.

b. A POSITA would not have been motivated to combine Cabrera, Schneider-PCT, and Whiting, because Schneider-PCT teaches away from file-level implementations.

488. As described earlier in the section titled *Schneider-PCT Deficiency #1: Schneider-PCT teaches away from any file-level implementations*, Schneider-PCT teaches away from file-level implementations, which Cabrera and Whiting rely on.

V. CONCLUSION

489. For the foregoing reasons, in my opinion, Claims 1-6 and 8-11 of the ’478 Patent and Claims 1-6, 8-20, and 23-31 of the ’348 Patent are not obvious in view of Kenley and Long and are not obvious in view of Cabrera, Schneider, and Whiting.

490. I reserve the right to modify and/or supplement this Declaration, to present further analyses, and/or to amend my opinions as more and/or other arguments and/or information becomes available as well as for any reason or for no reason.

491. I declare under penalty of perjury under the laws of the Unites States of America that the foregoing is true and correct.

February 28, 2024
Date

Zaydoon ("Jay") Jawadi