

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

DROPBOX, INC. AND MICROSOFT CORPORATION,

Petitioners,

v.

DATANET, LLC

Patent Owner.

U.S. Pat. No. 9,218,348

Inter Partes Review Case No. 2024-00079

PATENT OWNER RESPONSE

TABLE OF CONTENTS

I.	INTRODUCTION	1
II.	DESCRIPTION OF THE CHALLENGED PATENT	2
III.	LEVEL OF SKILL	4
IV.	WITHDRAWAL OF EXHIBITS 2001 AND RELATED EXHIBIT 2002	4
V.	ARGUMENT	5
A.	Response to Ground 1: Kenley in view of Long	5
1.	Overview of Kenley	5
2.	Overview of Long	5
3.	The Petition fails to show that Limitation 1.4 is disclosed or suggested by Kenley in view of Long	6
a)	Long Flushing the Memory Buffer Does Not Disclose Identifying the Presence of Any Alleged Archive File	7
b)	Long's Semaphoring the Memory Buffer Does Not Indicate Identifying the Presence of the Alleged Archive File	13
c)	The Proposed Combination with Kenley Does Not Cure the Above Deficiencies of Long	14
4.	The Petition fails to show that Limitation 1.6 is disclosed by Kenley in view of Long	15
5.	The Petition fails to show that Limitation 1.9 is disclosed by Kenley in view of Long	21
6.	The Petition fails to show that Limitation 1.10 is disclosed by Kenley in view of Long	24
7.	It would not have been obvious for a POSITA to combine Long and Kenley as proposed in Ground 1	27
a)	Long Teaches Away from a Combination with Higher-Level File Systems	29
b)	Combining Long with Kenley Would Change the Principle of Operation of Long or Kenley	30
B.	Response to Ground 2: Cabrera in view of Schneider and Whiting	31
1.	Overview of Cabrera	31

2. Overview of Schneider.....	31
3. Overview of Whiting.....	32
4. The Petition fails to show that Limitation 1.4 is disclosed or suggested by Ground 2’s proposed combination.....	32
a) The Staging File of Cabrera Does Not Correspond to the Claimed Archive File.....	34
b) The “Staging File” Is Only Disclosed in Conjunction with Figure 5, and Does Not Differ from the “Staging Area” – a Conglomeration of Data from the Local Disk	37
5. The Petition fails to show that Limitation 1.5 is disclosed or suggested by Ground 2’s Proposed Combination.....	41
a) Ground 2 Uses Cabrera’s Intermediate System to Meet Limitation 1.8.	41
b) Limitation 1.5 Requires Moving the Archive File to a Second Storage Location Responsive to a Second Event.....	42
c) Cabrera Is Silent on When or What Might Trigger Moving Data from the Intermediate System to Remote Storage.....	43
6. It would not have been obvious to combine Cabrera and Schneider as proposed, as altering Cabrera as proposed would not work, and it would also fundamentally alter the operation of Cabrera	46
a) Cabrera Is Fundamentally Periodic in Nature, and Making the Proposed Change Would Alter the Principle of Operation of Cabrera.....	46
b) The Proposed Combination Would Not Work in the System of Cabrera	48
7. Whiting Does Not Disclose or Suggest Updating a Database to Reflect Storage of an Archive File in Multiple Locations, and a POSITA Would Not Combine Whiting with Schneider and Cabrera	52
VI. CONCLUSION.....	56

LIST OF PATENT OWNER'S EXHIBITS

Exhibit	Description
2001	[WITHDRAWN] Declaration of Zaydoon (“Jay”) Jawadi in Support of Patent Owner’s Preliminary Response
2002	[WITHDRAWN] Curriculum Vitae of Zaydoon (“Jay”) Jawadi
2003	Computer Busses, Bill Buchanan, CRC Press, 2000, pp. 139-143
2004	The Multi-boot Configuration Handbook, Roderick W. Smith, Que, 2000
2005	Encyclopedia of Microcomputers, Volume 2 - Authoring Systems for Interactive Video to Compiler Design; Allen Kent, James G. Williams, and Rosalind Kent; Taylor & Francis; 1988
2006	PhoenixBIOS 4.0 Revision 6 User's Manual; Phoenix Technologies Ltd.; June 22, 2000
2007	Forensic Computing: A Practitioner's Guide; A. J. Sammes, Brian Jenkinson; Springer Science & Business Media; 2000
2008	Introduction to Software Engineering; Ronald J. Leach; CRC-Press; 2000
2009	PC Interrupts: A Programmer's Reference to BIOS, DOS, and Thirdparty Calls; Ralf Brown, James Kyle; Addison-Wesley; 1991
2010	Hard Disk Controller: the Disk Drive’s Brain and Body; James Jeppesen, Walt Allen, Steve Anderson, Michael Pils!; Infineon Technologies; 2001
2011	SCSI-2 Standard; ANSI; 1994
2012	ATA Draft Standard; ANSI
2013	Newton's Telecom Dictionary: The Official Dictionary of Telecommunications Networking and Internet; 16th Edition; Harry Newton, Ray Horak; Cmp Books; 2000

2014	U.S. Patent Publication No. 2009/0254594 to Laurion D. Burchall
2015	<i>Reserved</i>
2016	U.S. Patent Publication No. 2007/0226436 to Cheng et al.
2017	U.S. Patent Publication No. 2004/0003103 to Witt et al.
2018	Foundations of Computer Technology; A. John Anderson; 1994; Taylor & Francis
2019	Redline comparing Pet., 19-25 with EX1003, ¶¶49-58
2020	Declaration of Zaydoon (“Jay”) Jawadi in Support of Patent Owner’s Response
2021	Curriculum Vitae of Zaydoon (“Jay”) Jawadi
2022	Deposition of David Maier, Ph.D., July 18, 2024
2023	<i>Reserved</i>
2024	The International Dictionary of Data Communications; Robert A. Saigh; 1998; Glenlake Publishing Company; pp. 159, 234, 278

I. INTRODUCTION

Pursuant to 37 CFR § 42.120, Datamet, LLC (“Datamet” or “Patent Owner”) files this response to the Petition and the Institution Decision.

U.S. Patent No. 9,218,348 (the “’348 Patent”) relates to a method for archiving files. *See, e.g.*, EX1001-claim 1, Abstract. The ’348 Patent describes a novel method of backing up a computer—providing real-time, continuous backup even when the desired backup location is not available, and doing so with an imperceptible impact on system performance from a user’s point of view.

Ground 1, the proposed combination of Kenley and Long, fails to render the challenged claims obvious because it fails to disclose various limitations of the claims, including identifying the presence of the archive file in the temporary storage location responsive to the occurrence of a first event (Limitation 1.4), updating a database to indicate that the archive file is located in the temporary storage location (Limitation 1.6), or updating the database to indicate the archive file is located in the intermediate storage location (Limitation 1.9) or in the second storage location (Limitation 1.10). Further, a POSITA would not have combined Long with Kenley in the manner proposed, because Long is concerned only with hardware, specifically disk sectors and associated data within a disk sector, whereas Kenley is a file-based system.

Ground 2, the proposed combination of Schneider, Cabrera, and Whiting, fails to render the challenged claims obvious because it fails to disclose various limitations of the claims, including identifying the presence of the archive file in the temporary storage location responsive to the occurrence of a first event (Limitation 1.4), or transmitting the archive file to a second storage location responsive to a second event (Limitation 1.5), as mapped in the Petition. Further, it would not have been obvious to combine Schneider and Cabrera in the manner proposed, because the proposed backup trigger from Schneider is before a disk write, whereas the backup system of Cabrera backs up data once it has already been written by Cabrera's data producers. Further, Whiting fails to disclose or suggest updating a database to reflect storage of an archive file in multiple locations, as recited in Limitations 1.6, 1.9, and 1.10, and a POSITA would not have combined Whiting with Schneider and Cabrera in the manner proposed, since combining Whiting in such a fashion would constitute classic and impermissible hindsight.

II. DESCRIPTION OF THE CHALLENGED PATENT

The '348 Patent is directed to improvements to backup technology dating back to the late 1990s into the year 2000. During that time, backup programs were typically run according to a predetermined, fixed schedule, and due to slow network data transmission rates and limited storage availability, the backup 'jobs'

would typically be run at most every hour, but more likely once a day or once per week. *See* EX1001-2:17-30.

File modifications made in between backup operations would be lost if the computer crashed or the file was lost or corrupted, and the systems then generally required that backup storage devices were connected and available at the scheduled backup time, or the scheduled backup would fail. EX1001-2:17-30. Outdated backup files and data loss were the norm, with serious consequences. EX1001-1:48-53.

The '348 Patent claims an entirely new way of managing data protection and backups that fundamentally altered the way computer backups operated, providing near real-time backup operations even when backup storage devices are temporarily unavailable. EX1001-6:39-48. Datanet's claimed technology actively monitored a computer's operating system for, as one example, requests to save a modified file. EX1001-7:16-32. Upon detecting those operating system instructions, the technology initiates, in near real time, a backup operation to a local queue or buffer. *Id.* When a suitable backup storage device becomes available, the modified file contents are transmitted to the backup storage device for long-term storage. *Id.*

The backup approach disclosed and claimed in the '348 Patent, captures and archives contents of a working file temporally proximate to the time a computer's

operating system attempts to access and/or modify the file. This was a large step forward in the methodology of backup technology that is widely used today, which is why Petitioners are attempting to invalidate the '348 Patent.

III. LEVEL OF SKILL

Petitioners have provided a definition of a POSITA that such a person “would have held a bachelor’s degree in computer science, electrical engineering, computer engineering, or a related field, and one or more years of experience in the design and development of file management, restoration, and archival systems.”

Petition 6-7. For purposes of this proceeding only, Patent Owner does not dispute this definition of a Person of Ordinary Skill in the art.

IV. WITHDRAWAL OF EXHIBITS 2001 AND RELATED EXHIBIT 2002

According to Patent Trial and Appeal Board Consolidated Trial Practice Guide (Nov. 2019) (“Practice Guide”):

If a trial is instituted, a patent owner may choose not to rely on testimony submitted with the preliminary response. In that case, the patent owner should advise the Board of its decision, which can be done by affirmatively withdrawing the testimony in the patent owner response. If a patent owner withdraws a declaration submitted with its preliminary response, that declarant will usually not be subject to deposition on the withdrawn declaration.

Practice Guide § II(C), at 51.

Pursuant to this direction, and as indicated above in the List of Patent Owner’s Exhibits, Patent Owner hereby advises the Board that Patent Owner is withdrawing

the testimony of Zaydoon (“Jay”) Jawadi (EX2001) dated February 28, 2024, and related Curriculum Vitae in Exhibit 2002, that were submitted with Patent Owner’s Preliminary Response (Paper 8). Patent Owner submits a declaration of Zaydoon (“Jay”) Jawadi, dated August 2, 2024 herewith, as Exhibit 2020, and an accompanying Curriculum Vitae as Exhibit 2021.

V. ARGUMENT

A. Response to Ground 1: Kenley in view of Long

1. Overview of Kenley

Kenley (U.S. Patent No. 5,276,867) teaches a variation on the concept of hierarchical storage systems for files, noting a “hierarchical” storage server “emulates a large amount of high-speed secondary storage through use of a fixed amount of high-speed secondary storage as a cache for a large amount of slower tertiary storage.” EX1014-4:48-51. The stated goal for the system is “provid[ing] digital data storage systems having the speed and convenience of online data storage, and the economy and large capacity of offline storage.” EX1014-2:19-22.

2. Overview of Long

Long (U.S. Patent No. 5,325,519) is a hardware-related patent that relies on capturing system events and copies images of hard disk sectors for purposes of hard disk error recovery. EX1004-1:56-59, 4:40-42, 7:37-40. Long teaches that error detection and containment should be “confined at the lowest possible level”

and that errors at that level “are more easily detected than errors originating within modules that generate or transform data.” EX1004-3:1-50. Long does not concern files, but rather the sectors on the physical hard disk that contain binary data.

3. The Petition fails to show that Limitation 1.4 is disclosed or suggested by Kenley in view of Long

The combination of Kenley and Long does not disclose or suggest “the step of identifying the presence of the archive file in the temporary storage location responsive to the occurrence of a first event,” as recited in Limitation 1.4. Nothing in the Petition, the Petitioners’ expert declaration, nor in the Institution Decision points to any disclosure or suggestion in the proposed combination that discloses or suggests the recited “identifying presence of *the archive file* in the temporary storage location responsive to the occurrence of a first event.”

The Petition glosses over details of the challenged claims in its argument that “Long teaches identifying the presence of the *archived data* in the memory buffer” responsive to a first event. Petition 24. Limitation 1.4, however, recites identifying the presence of the specific *archive file*, and not just the presence of any data.

Long does not perform any identification of the markers (that is, the sector data that Long stores) stored in the memory buffer, which the Petition maps to the temporary storage location, *in response to* the alleged first events. The alleged “first events” mapped in the Petition result in Long moving (flushing) its memory buffer data from the memory buffer to the physical hard disk, but nothing in Long

discloses or suggests that, in preparing for or making such a move, Long identifies the presence of any particular marker data within the memory buffer. Therefore, Long-Kenley fails to disclose or suggest Limitation 1.4.

The Petition also points to Long's disclosure of "semaphoring," which, at best, indicates or signals the presence of some data in the memory buffer, but the mere presence of a semaphore does not teach or suggest *identifying the presence of* any particular data within the memory buffer. EX2020 ¶ 155.

a) Long Flushing the Memory Buffer Does Not Disclose Identifying the Presence of Any Alleged Archive File

The Petition cites to two disclosures in Long as being the recited "first event": "(i) the memory buffer filling up and (ii) the 'processing unit 14 requir[ing] memory used in the memory buffer for another purpose.'" Petition 24 (citing EX1004-7:47-53). The Petition then presents the attorney argument that "[e]ither event triggers the audit subsystem identifying the data's presence in the memory buffer as reflected by the fact that, responsive to either of these two events, this data is moved to the audit buffer." *Id.* at 24-25 (citing EX1004-8:20-23). The following quote from Long makes clear that the presence of particular marker data is not being identified but instead the buffer is merely "flush[ed]." "When another part of the computer system 10 needs to write data to the hard disk, it will flush the memory buffer that contains the audit markers to the physical disk." EX1004-8:20-23. The Long memory buffer holds a plurality of Long's markers. *See, e.g.,*

EX1004-Fig.2b. Flushing that buffer to disk does not entail identifying the presence of any of its constituent parts. EX2020 ¶ 158. Flushing does not entail identifying the presence of any particular archive file as required by Limitation 1.4.

After the above quote (EX1004-8:20-23), the Petition merely restates the erroneous conclusion, that either of these events would constitute a first event and that it “triggers identifying the presence of the data in the memory buffer.” Petition 25 (citing EX1003 ¶ 66). Limitation 1.4 requires “identifying the presence of the archive file,” not moving/flushing/dumping data or identifying the presence of a conglomeration of data, *i.e.*, the markers stored in Long’s memory buffer.

It is also important that the archive file takes its antecedent basis from Limitation 1.3, “creating an archive file from the operating file,” which operating file is “a system or user file.” EX1001-4:11. The archive file is created from an individual operating file. The archive file, further, is a file containing all of the data of the operating file or at least some of the data and references to the location of the remainder of the data. EX1001-4:12-16.

Limitation 1.4 does not simply recite identifying the presence of *data*, but rather identifying the presence of *the archive file*.

The Petition goes over how Long creates its markers, outlining Long’s Figure 3, which shows the marker header, the data, and the marker trailer. *See* EX1004-Fig. 3; Petition 25-26. The Petition states that “[t]he ‘audit marker 32 comprises a

marker header 34 and a marker trailer 36' where captured data 'is appended to the marker header 34.'" Petition 25 (quoting EX1004-5:30-34, Fig. 3). The Petition argues that that this is information that is used to determine the location of the marker. *Id.*

However, the existence of the marker data does not suggest that any identification of particular marker data occurs in response to any particular event in Long, let alone the two events that the Petition cites – (i) the memory buffer filling up, and (ii) the processing unit requiring the memory of the memory buffer for another purpose. Nothing in the Petition, the Petitioners' expert declaration, or Long suggests otherwise.

When Long "flush[es]" the data out of the memory buffer, nothing indicates that Long must or that it actually does identify the presence of any particular marker or data appended to any particular marker. All that happens is that the memory buffer – the entire memory buffer – is moved to the physical disk. EX1004-7:47-53. Nothing within the memory buffer is identified at this point, responsive to one of these events (or indeed at any other point in Long). This moving of the memory buffer contents would likely be done as a block copy in Long, *i.e.*, a copy of a set of memory without identifying the presence of particular markers within the memory. EX2020 ¶ 158. Long's audit header tracks the beginning and the end of its memory buffer, which buffer is stored contiguously in

memory. These features of Long enable a block copy of the full memory buffer from the first memory location through the last without any identification of the markers / data within the buffer. *Id.*

The memory buffer of Long holds multiple markers, as recognized in the Petition. Petition 25-26. The memory buffer holding multiple markers is also evident from the fact that the memory buffer of Long has its own audit header. EX1004-7:47-52 (When the memory buffer is full or the memory is otherwise required by the system, “the audit subsystem writes the data stored in the memory buffer to the end of the audit buffer 30 on disk (block 56) and updates the audit header (in the memory buffer) to reflect the changes (block 58).”). The Petition argues that the audit header stores information regarding markers. Petition 26. But critically, Long’s audit header only stores information on the first and the last markers in the buffer. EX2020 ¶ 159.

Table 1 from Long, which details the data stored in Long’s audit, as highlighted by the Petition at 31-32 is below:

TABLE 1

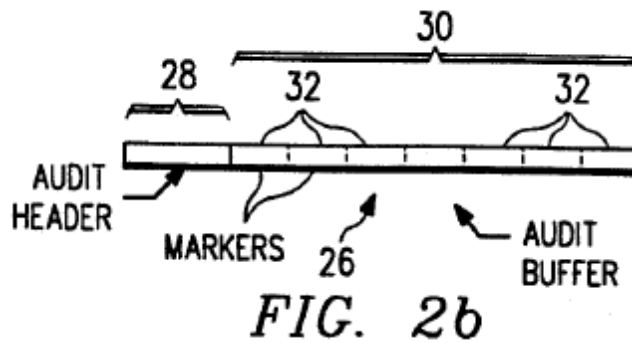
Audit Header Fields	
Data Label	Data Description
Ver	The version of the audit system that created the audit partition.
Name	A name to assist in determining that the audit partition is valid.
HdrSeqNum	Sequence number that is used every time the header is written to disk.
Time	The time the audit partition was created or reset (cleared out).
CloseErr	The last audit error. This is used to determining if a failure occurred and the system came down before the audit

TABLE 1-continued

Audit Header Fields	
Data Label	Data Description
	header was updated on the hard disk(s).
SeqNum	The next Archive disk sequence number to use.
QSeqNum	The last archive disk that contains a user quiescence marker.
Flush	A note to the system that the audit header needs to be written to disk at the next available opportunity (used in memory only).
LabelLoc	The physical location of the label on disk (audit header).
BufLoc	The physical location of the beginning of the audit buffer on disk.
CacheLoc	The current location of the audit buffer in the disk cache.
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.
CLocBytes	Current offset in the current sector that we are at (see CLoc).
MaxSectors	The number of sectors that make up the audit buffer. This does not include the first sector that contains the audit header.
MaxUsable	When the audit buffer gets to this point the archive system will start archiving.
MinUsable	When the audit buffer gets to this point the archive system will stop archiving.
UsedSectors	The number of sectors that have been used in the audit buffer.
CurSector	The logical sector number currently in use by the audit buffer.
FirstMarker	The contents of the first marker header.
FirstMarkerLoc	The location of the first marker in the audit buffer.
FirstMarkerOffset	The offset in the sector that contains the first marker.
FirstMarkerSector	The logical sector that contains the first marker.
LastMarker	The contents of the last marker header.
LastMarkerLoc	The last marker location in the audit buffer.
LastMarkerOffset	The offset in the sector that contains the last marker.

As is clear from Table 1 of Long, Long's audit header, as far as information on markers, stores only data for the first and the last markers. (FirstMarkerLoc, FirstMarkerOffset, FirstMarkerSector, LastMarker, LastMarkerLoc, LastMarkerOffset). Information for any other markers, such as, e.g., the second, third, or fourth markers in Long, is not stored in Long's audit header. *Id.*

The audit header and markers of Long are stored in a sequence, as can be seen, for example, in Figure 2b of Long:



Long's Figure 2b is disclosed in relation to the audit partition on the disk (EX1004-4:40-42), but it is just as descriptive of the memory buffer that stores the same sort of markers and audit headers. EX2020 ¶ 159. There is no disclosure in Long that the markers in the buffer memory, which are explicitly stored with an audit header (EX1004-7:47-53), would be stored any differently than they are stored in the audit buffer on the disk.

When the memory buffer is flushed / copied to the disk, Long's system does not need to identify the presence of particular markers, which are the contents of the memory buffer. This is because the Long audit header stores the information on the first and the last markers, with the remaining markers stored contiguously in memory therebetween. EX2020 ¶ 159. The data from the start to the end of the buffer is copied together to the physical disk and the data within the buffer does not need to be identified by the Long system. *Id.*

The contents of the memory buffer do not need to be and are not identified in order to flush the memory buffer, and the Petition is wrong to allege otherwise.

Long tracks the beginning and the end of the buffer, which is sufficient information for Long to move the full set of markers in the memory buffer, which are then appended to the end of the set of markers that are in the audit partition on the physical disk, without any further *identification* of the presence of the markers within the buffer as required by Limitation 1.4. Simply put, Long is not identifying the presence of markers, the constituent parts of the buffer, but merely block-copying the full buffer from one location to another. EX2020 ¶ 159. The Long memory buffer's audit header is then reset, and the information regarding the last marker (and the overall length of the memory buffer, which can be determined from the locations of the first and last markers) will be used to update the audit header of the audit partition. *Id.*

b) Long's Semaphoring the Memory Buffer Does Not Indicate Identifying the Presence of the Alleged Archive File

The Petition also cites to Long's disclosure regarding the memory buffer being semaphored. Petition 26 (quoting EX1004-8:18-20). The Petition argues that “[a] semaphore on the memory buffer indicates the presence of the data in the memory buffer to other processes.” *Id.* (citing EX1019-354 (defining “semaphore”)). As defined in EX1019, a semaphore is a “signal” or a “flag variable—an indicator” used to govern access to a shared system resource. EX1019-354.

A semaphore as used in the context of Long – having the memory buffer “semaphored” – simply means that there is a signal or flag-variable that indicates

whether the memory buffer is editable or locked. EX2020 ¶ 162. Even assuming that the memory buffer is only locked when there is data present in the memory buffer – which is not clear from Long’s limited disclosure of merely “having the memory buffer semaphored” and flushing the memory buffer to the physical disk (EX1004-8:19-23) – the presence of a semaphore would merely indicate that there is data present, not anything about the contents of the data. And nothing in Long concerning semaphores indicates the presence of any particular marker information as required by “identifying the presence of *the archive file*” responsive to any alleged event in this instance. All that would happen is that whatever write is being attempted to the memory buffer will be blocked until the prior data in the memory buffer is flushed, the buffer is therefore empty, and the semaphore signal changes from blocking writing of data to allowing writing of data. This semaphore disclosure has nothing to do with identifying the presence of any particular data at all, let alone in response to any event. EX2020 ¶ 163.

c) The Proposed Combination with Kenley Does Not Cure the Above Deficiencies of Long

After proceeding through the above disclosures of Long, the Petition erroneously argues in a conclusory fashion that using these mechanisms in a file-based system, such as Kenley, would “result in the audit subsystem identifying the presence of the copied/archive file in the memory buffer (temporary storage location) responsive to a first event.” Petition 23. This conclusion does not follow.

As detailed above, the cited portions of Long do not disclose or suggest identifying the presence of any particular data in the memory buffer, so merely porting these Long mechanisms over to Kenley – even assuming, *arguendo*, that this would be possible – does not disclose or suggest “identifying presence of the archive file in the temporary storage location responsive to the occurrence of a first event,” as recited in Limitation 1.4.

* * *

The proposed combination of Kenley and Long therefore fails to disclose or suggest Limitation 1.4, and Ground 1 fails as to claim 1. For at least the same reasons, the proposed combination of Kenley and Long fails to disclose or suggest Limitations 8.4, 12.4, 13.4, 14.4, and 15.4 and therefore fails as to claims 8, 12, 13, 14, and 15. Dependent claims 2-6 depend from claim 1, dependent claims 9-11 depend from claim 8, and dependent claims 16-20 and 23-31 depend from claim 15 and are patentable for at least the same reasons.

4. The Petition fails to show that Limitation 1.6 is disclosed by Kenley in view of Long

Limitation 1.6, as labeled by the Petition, recites “after storing the archive file in the temporary storage location, updating a database to indicate that the archive file is located in the temporary storage location.”

The Petition argues that the combination of Kenley and Long allegedly teaches Limitation 1.6, based on the allegation that “implement[ing] Kenley-Long such

that information on current locations of different backup files, including the relevant audit-header information taught by Long, was stored in the relational database catalog taught by Kenley.” Petition 32. This is the apparent basis for the combination meeting Limitation 1.6, the storing of Long’s alleged location information in Kenley’s database. This was confirmed by Petitioners’ expert. EX2022-58:14-59:2 (Q “So that’s the location information from Long audit header in the database of Kenley. Correct? A (Witness reviewing.) Yes . . .”). However, as discussed above in section 3, Long’s audit header stores location information only about the first and last marker. EX2020 ¶ 167.

The Petition begins its discussion of Limitation 1.6 with quotes from and characterizations of Kenley and its database, starting with a system catalog that is maintained by Kenley. Petition 30 (citing EX1014-3:17-28). Maintaining a system catalog does not teach storing the *location* of any archive files. EX2020 ¶ 168. The Petition also cites Kenley’s storing “backup catalogs (i.e., lists of all files copied to specific backup savesets).” Petition 30 (citing EX1014-6:35-40). Take note of how Kenley defines its backup catalog, using “i.e.” to clarify that the catalog is merely the “list[] of all files copied to specific backup savesets.” That is what Kenley’s backup catalog is, nothing more. Lists of files stored to specific “savesets” indicate *which* files are stored in a particular group but do not disclose the *location* of any particular archive file in its respective saveset. EX2020 ¶ 168.

The Petition then cites to Kenley’s disclosure of a “database 14 a catalog of all files backed up.” Petition 30 (citing EX1014-8:26-29). Again, this says nothing about *locations* of files being stored. As a last citation to Kenley for Limitation 1.6, the Petition cites a disclosure in the background of Kenley regarding storing data on offline media, such as tapes. *Id.* (citing EX1014-2:6-14). This is alleged in the Petition and by Petitioners’ Expert Declaration as showing that the failure to track locations was a problem that Kenley sought to solve. This disclosure, however, merely shows that there was a problem at the time with management of offline media. EX2020 ¶ 168. There is nothing in Kenley’s column 2, lines 1-14, that discloses or suggests updating a database to indicate that an archive file is located in the temporary storage location, as recited in Limitation 1.6.

The Kenley citations used in the Petition to meet Limitation 1.6 fail to disclose or suggest updating a database to indicate that an archive file is located in the temporary storage location. *Id.*

The Petition, after the four Kenley citations, then turns to Long. It argues that Long “teaches an audit header that is updated to reflect changes in the data’s storage location, including when the copied data is stored within the memory buffer (temporary storage location).” Petition 30 (citing EX1004-7:49-53). The Petition then argues that the audit header “contains information that is used to manage the audit buffer 30, archive media . . . and the general environment of the

audit subsystem.” *Id.* (citing EX1004-4:44-47, 4:57-60, 2:5-8). The Petition further argues “a number of fields” in Long’s audit header allegedly “specif[ies] the location of data managed by the audit subsystem,” including “CacheLoc” (specifying the location of the audit buffer itself), “BufLoc” (specifying the physical location of the beginning of the audit buffer), and “CLoc” (specifying the physical location of the last data in the buffer). Petition 30-31 (citing EX1004-4:61-5:24 (Table 1)).

As argued above in Section 3, the only location information stored in Long’s audit header is about the beginning and the end of the *buffer*; no information is store about the location of any individual markers in between because there are no such markers.

The Petition quotes Long as “disclos[ing] ‘construct[ing] the audit header’ for a marker after ‘[s]torage of the data in the memory buffer’ in the marker.” Petition 31 (citing EX1004-7:59-68). But this is actually a mistake in Long. Long does not construct an “audit header” for a marker, rather it constructs a marker header for a marker. EX2020 ¶ 171. This is clear from the language following the prior quote about constructing the “audit header.” The later description outlines the fields from the Long marker header, *i.e.*, “the type of marker (corresponding to the type of event), the markers’ CRC, the current system time, and the location that the sector data is from on the hard disk.” EX1004-7:65-68, Table 2 (showing that these fields

are part of Long's marker header, not audit header). Petitioners' expert admitted the mistake in Long. EX2022-69:21-70:6 (“[O]n previous readings I believe I’ve understood that this was talking about the initial construction of the audit header . . . [n]ow it would seem like this is also a possible typo in Long where it perhaps would have made more sense that this was talking about the marker header because it’s reciting elements of the marker header.”).

Long’s audit header is not created for each marker with location information, but is used once, at the beginning of Long’s audit buffer. EX2020 ¶ 171; EX1004-Fig. 2b. To be clear, Long’s audit header only stores location data for the first and last marker in the audit buffer.

The Petition then argues that “when the data is moved from the memory buffer (temporary storage) to the audit buffer, the audit header is updated ‘to reflect the changes.’” Petition 31 (citing EX1004-7:47-53). But the audit header is only updated for the last marker and does not track the location of any other markers, only the first and the last. EX2020 ¶ 172. The Petition further argues that “[c]onstant updating of this audit header to indicate the location of the data in the memory buffer and other locations, after storing the data in the respective locations, facilitates the audit subsystem’s accurate retrieval and movement of data for both backup and recovery.” Petition 31 (copied in the accompanying declaration, EX1003 ¶ 77). This argument fails, however, because the audit header

of Long does not indicate the location of the data within the memory buffer. The audit header of Long contains only location information for the first and the last marker. Any “updating,” whether constant or not, updates only this beginning and end of the audit buffer. EX2020 ¶ 172. Any updating of Long’s audit header would not indicate the location of the associated data in the memory buffer. *Id.*

The Petition goes on to say that “[t]hus, in view of these complementary teachings, a POSITA would have been motivated with a reasonable expectation of success to implement Kenley-Long such that information on current locations of different backup files, including the relevant audit-header information taught by Long, was stored in the relational database catalog taught by Kenley.” Petition 32; EX1003 ¶ 78. This is the Petition’s argument for Limitation 1.6 – using Long’s audit header information in Kenley’s database. But as discussed, Long’s audit header does not store the *location of the markers in the audit buffer* beyond the first and the last. EX2020 ¶ 173.

The Petition concludes its discussion of Limitation 1.6 with a statement that “as with other elements, when used to implement a file-based system such as Kenley’s, these database updates taught by Long would indicate that the copied file (archive file) is located in the memory buffer (temporary storage location).” Petition 33; EX1003 ¶ 79.

The alleged “database updates” taught by Long, as argued in the Petition, are not taught by Long at all, and updates to the Long audit headers do not keep track of the location of more than two markers, the beginning and the end of the audit buffer (the first and the last marker). Therefore, the combination of Long’s audit header information with Kenley’s database would not allow the Kenley database to track the location of the alleged “archive files.” EX2020 ¶ 174.

The Kenley-Long combination therefore does not disclose or suggest “after storing the archive file in the temporary storage location, updating a database to indicate that the archive file is located in the temporary storage location,” as recited in Limitation 1.6. EX2020 ¶ 175. The proposed combination of Kenley and Long therefore fails to disclose or suggest Limitation 1.6, and Ground 1 fails as to claim 1. For at least the same reasons, the proposed combination of Kenley and Long fails to disclose or suggest Limitation 8.6 and therefore fails as to claim 8. Dependent claims 2-6 depend from claim 1 and dependent claims 9-11 depend from claim 8 and are patentable for at least the same reasons.

5. The Petition fails to show that Limitation 1.9 is disclosed by Kenley in view of Long

As discussed above in Section 4, Kenley-Long fails to meet Limitation 1.6. The same reasoning is used in the Petition and accompanying declaration for Limitation 1.9 as for Limitation 1.6. For the same reasons as in Section 4, the Kenley-Long combination does not disclose or suggest “updating the database to indicate the

archive file is located in the intermediate storage location,” as recited in Limitation 1.9.

The Petition starts its Limitation 1.9 discussion by stating that “Long teaches updating its audit header to indicate the copied data (archive data) is located in the audit buffer (intermediate storage): after ‘writ[ing] the data stored in the memory buffer to the end of the audit buffer 30 on disk’ Long specifies ‘determ[ing] the audit header (in the memory buffer) to reflect the changes.’” Petition 36 (copied in EX1003 ¶ 89) (citing EX1004-7:47-53, Fig. 5). This Long statement only shows Long updating the audit header in the memory buffer (the alleged temporary storage location), when data is moved to the physical disk to indicate that the memory buffer being empty. Nothing in this statement relates to the audit buffer on the disk, the alleged intermediate storage location of the Kenley-Long combination. EX2020 ¶ 179.

The Petition goes on to argue that “[t]he audit header is updated to indicate ‘[t]he physical location of the beginning of the audit buffer on disk.’” Petition 37 (copied in EX1003 ¶ 89) (citing EX1004-4:61-5:24 (Table 1) (defining “BufLoc”)). This is not correct. Although there is a BufLoc variable in the audit header on the disk, which audit header keeps track of the beginning of the audit buffer, the BufLoc variable is not updated when data is added to the end of the audit buffer on the disk (the alleged intermediate location) from the memory buffer

(the alleged temporary storage location), which data would be added to the end of the disk's audit buffer. EX2020 ¶ 180. When data is added to the end of an audit buffer, the "BufLoc" that tracks the beginning would not be updated. *Id.*

The Petition then states, "[a]s discussed above for Element [1.6], Kenley teaches a complementary relational database with a backup catalog containing, e.g., 'lists of all files copied to specific backup savesets.'" Petition 37 (copied in EX1003 ¶ 90) (citing EX1014-6:35-40). But, as discussed above in Section 4, Kenley's disclosure related to lists of files copied to specific backup savesets does not teach tracking the *location* of particular files, which are argued as the alleged "archive files."

The Petition then argues that Kenley-Long would meet Limitation 1.9 for the same reason it allegedly meets Limitation 1.6. But, as discussed above in Section 4, the Kenley-Long combination fails to meet Limitation 1.6 and, for at least the same reasons, it fails to meet Limitation 1.9.

As a result, Kenley-Long does not disclose or suggest "updating the database to indicate the archive file is located in the intermediate storage location" as recited in Limitation 1.9. EX2020 ¶¶ 181-183. The proposed combination of Kenley and Long therefore fails to disclose or suggest Limitation 1.9, and Ground 1 fails as to claim 1. For at least the same reasons, the proposed combination of Kenley and Long fails to disclose or suggest Limitation 8.8, which the Petition maps to the

same disclosure as 1.9 (Petition at 47), and therefore fails as to claim 8. Dependent claims 2-6 depend from claim 1 and dependent claims 9-11 depend from claim 8 and are patentable for at least the same reasons.

6. The Petition fails to show that Limitation 1.10 is disclosed by Kenley in view of Long

As discussed above in Sections 4 and 5, Kenley-Long fails to meet Limitations 1.6 and 1.9. The same reasoning is used in the Petition and accompanying declaration for Limitation 1.10 as for Limitations 1.6 and 1.9. For the same reasons as in Sections 4 and 5, the Kenley-Long combination does not disclose or suggest “after moving the archive file to the second storage location, updating the database to indicate that the archive file is located in the second storage location,” as recited in Limitation 1.10.

The Petition argues that “Long teaches updating its audit header to indicate that the copied/archived data is located in the removable archive media (second storage location), after the data has been moved to that location.” Petition 38; EX1003 ¶ 92 (citing EX1004-9:7-9, Fig. 6). But the cited portion of Long merely discusses that the audit header of the physical disk removes the reference to whatever data is moved at the end of the audit buffer. The Long citation does not show any sort of database update to indicate that any data is located in the archive media. EX2020 ¶ 186.

The Petition argues that Long’s audit header “includes fields specifying the location of data in the archive media that are ‘used to manage the . . . archive media.’” Petition 39; EX1003 ¶ 93 (citing EX1004-4:44-47). The disclosure in column 4 of Long mentions that the audit header is used to manage the archive media, but the Long citation related to the audit header or the fields pointed to on Petition pages 39 (which are used verbatim in paragraph 93 of the accompanying declaration) do not disclose or suggest using Long’s audit header to keep track of the location of markers of Long in the archive disk. The Petition (and declaration) cite to the “next Archive disk sequence number to use” and the “last archive disk that contains a user quiescence marker.” Petition 39; EX1003 ¶ 93 (citing EX1004-4:61-5:24 (Table 1)). These two fields, a sequence number or the last disk that contains a user quiescence marker, do not indicate the location of any markers of Long. EX2020 ¶ 187. None of the Long citations disclose or suggest that Long updates any of its audit headers to indicate the archive file is located in the second storage location, as recited in Limitation 1.10.

The Petition then states that “[a]s discussed above for Elements [1.6] and [1.9], Kenley teaches a database with a backup catalog containing, e.g., ‘lists of all files copied to specific backup savesets.’” Petition 40 (citing EX1014-6:35-40).

However, a POSITA would not understand the Kenley disclosure related to lists of

files copied to specific backup savesets to disclose tracking the location of particular files, alleged as “archive files.” EX2020 ¶ 188.

The Petition makes clear that the entire basis for Limitation 1.10 – as well as Limitations 1.6 and 1.9 – is that “a POSITA would have understood the audit header as a database indicating data locations.” Petition 40. As outlined at length herein, a POSITA would not understand Long’s audit header to disclose or suggest any such database indicating data locations. Long’s audit header only tracks the beginning and the end of its associated audit buffer, not the locations of data therein.

For the same reasons that it fails to meet Limitation 1.6 and 1.9, as discussed herein and in Sections 4 and 5, the combination of Kenley and Long fails to meet Limitation 1.10. EX2020 ¶ 189.

Kenley-Long therefore does not disclose or suggest “after moving the archive file to the second storage location, updating the database to indicate the archive file is located in the second storage location” as recited in Limitation 1.10. The proposed combination of Kenley and Long therefore fails to disclose or suggest Limitation 1.10, and Ground 1 fails as to claim 1. For at least the same reasons, the proposed combination of Kenley and Long fails to disclose or suggest Limitation 8.10, which the Petition maps to the same disclosure as 1.10 (Petition at 48), and therefore fails as to claim 8. Dependent claims 2-6 depend from claim 1 and

dependent claims 9-11 depend from claim 8 and are patentable for at least the same reasons.

7. It would not have been obvious for a POSITA to combine Long and Kenley as proposed in Ground 1

In addition to the failure of the Kenley-Long Ground 1 combination to teach or suggest features of the challenged claims, as discussed in Sections 3, 4, 5, and 6, Ground 1 fails because it would not have been obvious for a POSITA to combine Long and Kenley as proposed in Ground 1.

While Kenley and Long each discuss techniques that relate generally to the handling of data, the two references are from different types of systems that are mutually exclusive and incompatible. *See* EX2020 ¶¶ 57-63. Therefore, it would not have been obvious to combine Long – a hardware system that deals with sector writes to a physical hard disk, as well as other hardware events, with a file-system-level reference like Kenley.

Long relies on capturing images of hard disk sectors for purposes of hard disk error recovery. EX1004-1:56-59, 7:37-40, Fig. 5; EX2020 ¶ 132. Long never addresses the concept of “files” or “file backup,” which Petitioners’ expert admits. EX2022-44:11-12 (“Long does not contain the word ‘file’ nor the word ‘files’ to my knowledge.”). The low (near the hardware) level implementation of Long would not and could not recognize the concept of files. EX2020 ¶ 136. Petitioners’ expert admitted that Long was concerned with hardware. EX2022-45:18-24

(Long’s methods are “obviously in the context of hardware because they’re talking about disk drives”), 47:19-22 (“Long is... concerned with the hardware level, certainly. It is interposing itself between application programs and that hardware level.”). Of course, to fit with Petitioners’ theory in this case, Petitioners’ expert attempted to couch Long as being more broadly applicable. EX2022-45:22-24, 47:23-48:2.

Kenley, on the other hand, teaches a variation on the concept of hierarchical storage system for files. EX1014-2:33-36, 2:44-48; EX2020 ¶ 140. Kenley is completely dependent on a file system-level implementation and works by staging-in and staging-out particular files through the various storage tiers. EX1014-5:16-20, 5:48-6:18.

File systems and physical disks/sectors exist at separate conceptual levels in data storage systems, with hardware agnostic file systems built on top of hardware specific physical disk/sector drivers. *See* EX2020 ¶¶ 141-149.

As has long been recognized in the art, techniques and solutions for managing data at the file system-level are rarely applicable to the disk/sector-level, and vice versa. *Id.* Despite the well understood challenges of mixing file system and disk/sector-level techniques, Petitioners’ challenge amounts to an unsupported conclusion that the mere existence of the two different conceptual systems amounts to a motivation to combine their teachings.

a) Long Teaches Away from a Combination with Higher-Level File Systems

A POSITA would have understood Long as teaching that detecting access to a storage device should occur at as low a level as possible. For example, Long teaches that “[t]he detection of an error or its symptoms during the normal operation of the system is the cornerstone to fault tolerant architectures.” EX1004-3:3-5. Long teaches that its error detection methods are for “[e]rrors occurring in [low-level] data-storage components, such as registers and memory” because they are far easier to detect than “errors originating within modules that generate or transform data” (as would be the case with file system level modules). EX1004-3:7-11; EX2020 ¶¶ 123, 133-136.

These disclosures of Long specifically teach away from trying to implement Long’s error detection capabilities at a level used to “generate or transform data” such as the file system level at which Kenley and the Datanet challenged claims are directed.

Furthermore, Long specifically teaches that errors must be “confined to the lowest possible level.” EX1004-3:27-34. It teaches accomplishing this through low-level “processing circuitry [] to detect access to the mass storage device.” EX1004-1:65-2:2. Such low-level processing as contemplated in Long is incompatible with file-based implementation of Kenley, and the Petition has not explained how or

why a POSITA would reengineer to combine the function-critical low-level implementation in Long into the file-based process of Kenley.

b) Combining Long with Kenley Would Change the Principle of Operation of Long or Kenley

The proposed combination would necessarily change the principle of operation of either reference, and therefore a POSITA would not have found it obvious to combine Long with Kenley. EX2020 ¶¶ 148-152. As discussed above, Kenley and Long operate at fundamentally different levels, Kenley at the file level and Long at the hardware level. Combining Kenley with Long would require reengineering Kenley, Long, or both references, which would be a significant undertaking. EX2020 ¶¶ 148, 151; *see also* M.P.E.P. § 2143.01 (“If the proposed modification or combination of the prior art would change the principle of operation of the prior art invention being modified, then the teachings of the references are not sufficient to render the claims *prima facie* obvious.”) (citing *In re Ratti*, 270 F.2d 810, 813 (CCPA 1959)).

Therefore, it would not have been obvious to combine Long and Kenley as proposed in Ground 1 of the Petition, and Ground 1 fails as to all challenged claims.

B. Response to Ground 2: Cabrera in view of Schneider and Whiting**1. Overview of Cabrera**

Cabrera (U.S. Patent No. 5,953,729) teaches a periodic system and method for archiving or backing up data produced from one or more data producers. EX1005-Abstract, Fig. 2, 6:58-8:49. The data in Cabrera is copied from local storage to a staging area, in which it is temporarily stored, and then transferred to remote storage. EX1005-7:3-7, 7:47-49.

Cabrera is concerned with periodic backup or archiving of data, either after a certain time delay or when sufficient data is ready for backup to fit onto a remote storage media of a specific size. EX1005-10:45-55. Cabrera is most concerned with the space required for the staging area, which stages data before it is transferred to remote storage. *See* EX1005-2:11-51 (discussing problems created by copying information to a staging area), 2:55-64 (discussing the Cabrera solution to minimize storage space required and administrative burden, using “sparse file technology”).

2. Overview of Schneider

Schneider (International Patent Publication No. WO 99/12101) discloses reverting a disk drive to an earlier point in time. EX1020-Abstract. Schneider teaches “changes made to the drive are saved in a circular history buffer which includes the old data, the time it was replaced by new data, and the original location of the data.” *Id.* “The circular history buffer may also be implemented by

saving new data elements into new locations and leaving the old data elements in their original locations.” *Id.* “The disk drive is reverted to an earlier point in time by replacing the new data element with the original data elements retrieved from the history buffer” *Id.*

Schneider teaches a separate and alternative file-level embodiment, noting “[a]lthough the present invention at the file level is possible it may be more difficult to implement, prove correct, and isolate as a subsystem.” EX1020-24:20-22.

3. Overview of Whiting

Whiting (U.S. Patent No. 5,778,395) discloses a system for backing up files from disk volumes to a common random-access backup storage means. EX1006-Abstract. Whiting stores all its backup data to a single backup disk, backup storage means 101 on network file server 100. EX1006-7:13-17.

4. The Petition fails to show that Limitation 1.4 is disclosed or suggested by Ground 2’s proposed combination

Ground 2’s proposed combination does not disclose or suggest “identifying presence of the archive file in the temporary storage location responsive to the occurrence of a first event,” as recited in Limitation 1.4. As discussed above in Section A.3.a, the archive file takes its antecedent basis from Limitation 1.3, *i.e.*, it is a particular archive file created from an operating file.

The Petition points to three aspects of Cabrera for this Limitation, none of which discloses or suggests identifying any particular archive file in response to an event, as recited in the limitation.

First, the Petition cites to language that Cabrera “monitor[s] the ‘amount of data in the staging area . . . in order to identify when a backup or archive session should be initiated.’” Petition 69 (citing EX1005-3:45-47). This has nothing to do with *identifying* any archive file in the temporary storage location. This concerns merely monitoring the total amount of data in the alleged temporary storage location. A POSITA would not think that monitoring the total amount of data would require or even suggest identifying a particular archive file. EX2020 ¶ 194.

The Petition then cites to a general disclosure in Cabrera that the “‘event monitor 152’ . . . ‘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.’” Petition 69 (citing EX1005-13:1-10). This is a generic statement about potential for monitoring and triggering events but has nothing to do with the temporary storage location at all, let alone identifying the archive file in the temporary storage location. EX2020 ¶ 195.

The Petition then states the following:

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.” *Id.*, 11:22-27. A POSITA would have understood that evaluating whether to append more

data to the staging file necessarily requires identifying the presence of the data in the staging file, and that this identifying step would occur responsive to the triggering event. EX1003, ¶165.

Petition 69. This allegation (which is copied by Petitioners' expert, Dr. Maier – both the quote and the conclusion), rests on the presumption that appending data to the staging file requires identifying the presence of specific data in the staging file. But this does not follow. As discussed in further depth below, Cabrera's "staging file" is a conglomeration, a set of data from multiple files, and the parts of this staging file do not need to be identified to append data thereto. EX2020 ¶ 196.

a) The Staging File of Cabrera Does Not Correspond to the Claimed Archive File

The staging file of Cabrera is a conglomeration of data. It is not the individual archive file recited in the challenged claim, which is created from an operating file, *i.e.*, an individual computer file on which an operation is performed. EX1001-8:47-54; EX2020 ¶ 197.

Cabrera, from its most general description of its systems, discloses the staging of "data" before it is transferred to remote storage. EX1005-1:9-11 ("[T]he present invention relates to systems and methods for temporarily storing or staging data prior to its transfer to remote storage."). The basis of Cabrera is systems that require access to files "twenty-four hours a day, seven days a week," such that you cannot "prevent access to a particular file during the archive or backup procedure." EX1005-1:64-2:3. In systems with such a requirement, "creating backup or archive

copies of information” can be challenging, but one method, which is what is employed by Cabrera, is to “duplicate the information that will be backed up or archived and ‘stage’ the information in a temporary storage area.” EX1005-2:3-9. The “*information*” so staged “may then be copied from the staging area and sent to backup or archive storage.” EX1005-2:9-10.

This is the basis for Cabrera, to be deployed in a system that requires access to files at all times, such that access to files cannot be blocked during the backup process, and so files are first copied to a staging area and subsequently copied to archival / backup. EX2020 ¶ 198. The staging area contains a set of data, which constitutes copies of many files. The background of Cabrera discusses utilizing archive media of a defined size and collecting sufficient information to completely fill this archive media before archiving the data. EX1005-2:35-40. This collection is the staging area. EX1005-2:41-43.¹

Cabrera’s description of maintaining data in the staging area and appending data when more is available is as follows:

¹ Though not relevant to the instant challenge, Cabrera is most concerned with sparse file technology to minimize the space needed for the staging area. *See, e.g.*, EX1005-2:11-28 (need to minimize storage space), 3:8-35, Title (“Using Sparse File Technology to Stage Data that Will then Be Stored in Remote Storage”).

The amount of data in the staging area may be monitored in order to identify when a backup or archive session should be initiated. In the alternative, the time since last backup or archive may be monitored and a session initiated when a particular time has elapsed. If additional data becomes available in local storage prior to the time that an archive or backup session is initiated, such data can be [sic] appended to the data already stored in the staging area.

EX1005-3:45-53. This is a discussion of “data” in the staging area – not a particular file that is the staging file. EX2020 ¶ 199. The discussion of appending data if more is available in the local storage area before the archive file is about adding “data” to the conglomeration of data within the staging area. This is not about any particular archive file created from an operating file that is stored as any sort of “staging file” in the staging area. It is *all the data* that has been staged, waiting either for sufficient data in the staging area or sufficient time to elapse such that an archive process should be started. *Id.*

Cabrera discloses that:

When a first event occurs, indicating that data local storage area should be transferred to a staging area, archive system 66 will move an appropriate amount of data from the local storage area, such as data file 64, to a staging storage area which is adapted for temporarily storing the data. Thus, embodiments within the present invention may comprise means for moving data from a local storage area used for data storage by a data producing service to a staging storage area used for temporarily staging the data.

EX1005-7:3-11. Any “means” for moving the data may be used, including copying the data or using direct memory transfer. EX1005-7:11-17. However, a POSITA

would know that “moving” data does not disclose or suggest identifying the constituent parts within that data. EX2020 ¶¶ 158-159.

b) The “Staging File” Is Only Disclosed in Conjunction with Figure 5, and Does Not Differ from the “Staging Area” – a Conglomeration of Data from the Local Disk

Figure 5 of Cabrera is discussed in the specification from column 10, line 37 and continuing on to column 12, line 17. In the discussion of Figure 5, Cabrera’s description of the staging area shifts from discussing it as an “area” with “data” to a “staging file.” This is not a separate embodiment or any actual shift in the underlying mechanism but merely a change in nomenclature. Petitioners’ own expert equates the staging area and staging file in his declaration. EX1003 ¶ 162 (“The staging file or staging area ‘is adapted for temporarily storing the data.’”).

This “staging file” of Cabrera is a conglomeration of data, not any specific file that has been copied for back up. EX2020 ¶ 200. It is a load of data waiting to be backed-up to archival media, like an optical disc. *Id.* On cross examination, Petitioners’ expert would not admit to this fact, stating that this detail was unimportant for his report. EX2022-79:16-82:21 (“You know, for purposes of my report, the really important part is that Cabrera teaches data moving to a temporary -- a first temporary staging area before moving on to remote storage.”). But the actual details matter here; since the staging file of Cabrera is a conglomeration of data and does not correspond to an archive file, appending data to Cabrera’s

staging file does not result in identifying the presence of an *archive file*, as recited in Limitation 1.4. Appending data to the whole staging file in Cabrera does not cause identifying the archive file within the staging file / temporary storage location which archive file is created from an operating file.

To make it clear that there is no change in mechanism from the remainder of Cabrera, the staging area discussion of Figure 5 uses the same language as earlier in the Cabrera specification:

In FIG. 5, the method begins with decision block 124 which identifies whether sufficient data resides in local storage for staging to a staging area. If sufficient data does not reside in local storage for staging, the system waits for a given period of time illustrated in FIG. 5 by time delay 126, and then rechecks the amount of data in local storage.

EX1005-10:39-45. This makes clear that Figure 5 of Cabrera is, again, talking about a staging area with a conglomeration of data from various files from local storage, just as the staging area is discussed throughout the specification. EX2020 ¶ 201.

Multiple subsequent mentions of the staging area in the Cabrera discussion of Figure 5 use the term “staging file.” *See, e.g.*, EX1005-10:56-61 (after triggering event, “execution proceeds to step 128 where data is copied from local storage to the staging file”). This disclosure confirms that data, conglomerated, is copied to the staging area, referred to as the staging file. EX2020 ¶ 202. Data, *i.e.*, a conglomeration, is copied to *the* staging file. The staging file is a bunch of data and

not “a” staging file that corresponds to any particular file from the local disc of Cabrera. *Id.*

In discussing the mechanisms of moving or copying the data, Cabrera interchangeably uses the staging area and staging file. EX1005-10:61-11:9.

The primary disclosure of Cabrera relied upon by the Petition, Petitioners’ expert, and the Institution Decision for Limitation 1.4 is as follows:

Thus, decision block 130 determines whether it is time to establish a remote storage connection. If not, then execution proceeds to decision block 138 where a determination is made as to whether more data should be appended to the staging file. This determination may be made based on any triggering event, as previously explained in conjunction with decision block 124 and decision block 130. **If more data that should be appended to the staging file exists, then the data is appended in step 140.** In either case, execution proceeds back to decision block 130 in order to wait for the occurrence of the second triggering event that will initiate connection to remote storage.

EX1005-11:20-31 (emphasis added). As the Petition argued, “[a] POSITA would have understood that evaluating whether to append more data to the staging file necessarily requires identifying the presence of the data in the staging file, and that this identifying step would occur responsive to the triggering event.” Petition 69 (citing EX1003 ¶ 165). Paragraph 165 of the Maier declaration recites this same quote from the Petition (with the same emphasis) in a conclusory fashion, adding nothing. EX1003 ¶ 165. The Institution Decision relied on the same logic. Institution Decision 37-38.

But even accepting, for the sake of argument, the logic that appending data to a file requires identifying that file, Cabrera does not disclose that the staging file is the archive file, and it is not. Cabrera discloses appending to – and arguably identifying– the *staging file* (not a specific archive file), which is the whole set of data that is waiting to be backed up to remote storage, to see if more data from local storage should be appended to the staging file before beginning the backup process. EX2020 ¶ 204. When data is appended to the “staging file,” it has nothing to do with any alleged “archive file” that is derived from an original “operating file,” as required by Limitation 1.4. *Id.* Cabrera does not identify the presence of the archive file within the staging area / staging file (temporary storage location).

Hence, even if one must identify a file to append something thereto, all that is identified in Cabrera is the entire staging file (or the end of that file) – all the data waiting to be backed up until a sufficient amount is collected (or a sufficient time has elapsed). This is not “identifying presence of *the archive file* in the temporary storage location responsive to the occurrence of a first event,” as recited in Limitation 1.4. The staging file is not the “archive file” created from the “operating file” as recited in the challenged claim. EX2020 ¶¶ 205-206.

* * *

The proposed combination of Schneider, Cabrera, and Whiting therefore fails to disclose or suggest Limitation 1.4, and Ground 2 fails as to claim 1. For at least the

same reasons, the proposed combination of Schneider, Cabrera, and Whiting fails to disclose or suggest Limitations 8.4, 12.4, 13.4, 14.4, and 15.4 and therefore fails as to claims 8, 12, 13, 14, and 15. Dependent claims 2-6 depend from claim 1, dependent claims 9-11 depend from claim 8, and dependent claims 16-20 and 23-31 depend from claim 15 and are patentable for at least the same reasons.

5. The Petition fails to show that Limitation 1.5 is disclosed or suggested by Ground 2's Proposed Combination

Limitation 1.5 recites “transmitting the archive file to a second storage location responsive to a second event, the second storage location being . . . a permanent storage location.” The proposed combination of Cabrera, Schneider, and Whiting fails to disclose or suggest Limitation 1.5.

a) Ground 2 Uses Cabrera's Intermediate System to Meet Limitation 1.8

It is important to note that, in order to meet other limitations of the challenged claims, the Petition utilizes the scant disclosure of Cabrera regarding “an intermediate system or subsystem.” *See* Petition 74 (regarding Limitation 1.8). This disclosure of Cabrera is used to meet Limitation 1.8's recitation of moving the archive file from the temporary storage location to “an intermediate storage location.”

Ground 2, as mapped by the Petition, utilizes the following disclosure of Cabrera

By way of example, and not limitation, in FIG. 2 such means is illustrated by arrows 78, 80, and 82 which illustrate data being moved from sparse file 76 and delivered to remote storage communication infrastructure 84. Such a means may be implemented by any mechanism capable of retrieving data from sparse file 76 and either directly delivering the data to remote storage 62 or to an intermediate system or subsystem which will, in turn, deliver the appropriate data to remote storage.

EX1005-7:52-60 (cited by Petition 73). The Petition also cites to Figure 2 and column 8, lines 5 to 11, which discusses the remote storage infrastructure, without reference to the intermediate system or subsystem.

b) Limitation 1.5 Requires Moving the Archive File to a Second Storage Location Responsive to a Second Event

In mapping Limitation 1.5 in Ground 2, the Petition points to Cabrera's teaching that "[w]hen the second triggering event occurs and it is time to establish a connection to remote storage . . . the data should be transferred from the staging file to remote storage." EX1005-11:46-48, Fig. 5 (cited by Petition 69-70). The Petition also points to potential triggers that could constitute this alleged second event.

But, even if the Petition's mapping of the second event and the second storage location are all correct, at most, this second event triggers the movement of the archive file from the staging area / staging file to remote storage in Cabrera's system when there is no intermediate system or subsystem. EX2020 ¶ 212. The relied-upon disclosure of Cabrera for Limitation 1.5 completely ignores Cabrera's

intermediate system or subsystem, which Ground 2 requires to meet other limitations of the challenged claims.

Nothing in the Petition, Petitioners' expert declaration, or Cabrera itself discusses what the mechanism is for moving data from Cabrera's intermediate location to the remote storage. There is simply no disclosure of any trigger or alleged second event in Cabrera that could meet the requirements of Limitation 1.5, "moving the archive file to a second storage location *responsive to a second event*," which second location the Petition maps to the second permanent storage location. Petition 69-71 (citing EX1005-11:46-48; FIG. 5) ("[w]hen the second triggering event occurs and it is time to establish *a connection to remote storage ...* the data should be transferred from the staging file to *remote storage*."); 6:61-62 (referencing "backup or archiving storage device," which Cabrera states is remote storage in the following phrase); EX1005-11:28-31 (connection to "remote storage"); annotated Fig. 5 (highlighting "transfer data from staging file to remote storage"); EX2020 ¶ 213. Cabrera does not disclose or suggest a "second event" that results in the data moving *from* the intermediate system to the remote storage. *Id.*

c) Cabrera Is Silent on When or What Might Trigger Moving Data from the Intermediate System to Remote Storage

There is scant disclosure in Cabrera of the intermediate system. It is only mentioned in three parts of Cabrera. The first, referenced above, is in column 7, where Cabrera simply states that the mechanism for moving data could be "any

mechanism capable of retrieving data from sparse file 76 and either directly delivering the data to remote storage 62 or to an intermediate system or subsystem which will, in turn, deliver the appropriate data to remote storage.” EX1005-7:56-60. Nothing in this disclosure gives any indication of when data would move from the intermediate system or subsystem to remote storage or, most importantly for the instant challenge, what, if any, alleged “second event” would trigger such a move. EX2020 ¶ 214. The Petition and Petitioners’ expert declaration are entirely silent on these aspects of the proposed combination of Ground 2.

There are only two other mentions of the intermediate system in Cabrera, neither of which is cited by the Petition or Petitioners’ expert declaration, but which are addressed herein for completeness. The mechanism to transfer data “may be nothing more than writing data to a local disk or other storage device or this may require transferring data over various networks or via various computer systems or other intermediate devices to remote storage.” EX1005-11:49-55. And “remote storage 158” may comprise a wide variety of storage mechanisms including, among others, “a remote storage device accessed via an intermediary or other intermediary computer device.” EX1005-13:62-67 (as corrected by Certificate of Correction).

None of the three, brief, disclosures regarding Cabrera’s intermediate system give any detail about how or when data would be moved from the intermediate

system to the remote storage. EX2020 ¶ 216. For purposes of the instant challenge, none of them provide any detail about what, if anything, might trigger such a move such that it could even be mapped to the recited “second event” of Limitation 1.5.

Id. And the Petition is completely silent on this point, with no help from the expert, who cites the same Cabrera disclosures as the Petition with respect to Limitation 1.5. EX1003 ¶ 166.

Therefore, the proposed combination of Ground 2 – Cabrera, Schneider, and Whiting – fails to disclose or suggest “moving the archive file to a second storage location responsive to a second event, the second storage location being . . . a permanent storage location” as recited in Limitation 1.5, and as Ground 2 attempts to map these locations, and Ground 2 fails for this additional reason as to claim 1. For at least the same reasons, the proposed combination of Schneider, Cabrera, and Whiting fails to disclose or suggest Limitations 8.5, 12.5, 13.5, 14.5, and 15.5 and therefore fails as to claims 8, 12, 13, 14, and 15. Dependent claims 2-6 depend from claim 1, dependent claims 9-11 depend from claim 8, and dependent claims 16-20 and 23-31 depend from claim 15 and are patentable for at least the same reasons. EX2020 ¶¶ 217-218.

6. It would not have been obvious to combine Cabrera and Schneider as proposed, as altering Cabrera as proposed would not work, and it would also fundamentally alter the operation of Cabrera

a) Cabrera Is Fundamentally Periodic in Nature, and Making the Proposed Change Would Alter the Principle of Operation of Cabrera

The entire premise of Cabrera is a periodic backing up of data. EX2020 ¶ 219.

In Cabrera, data is backed up from the local storage area to the staging area “when a defined amount of data has been stored locally or when a particular time has elapsed.” EX1005-3:40-41. This is the central nature of Cabrera, periodically backing up data to the staging area and then periodically backing up the data from the staging area to remote storage.

As stated by Cabrera,

The amount of data in the staging area may be monitored in order to identify when a backup or archive session should be initiated. In the alternative, the time since last backup or archive may be monitored and a Session initiated when a particular time has elapsed.

EX1005-3:45-50. This periodic nature is a central feature of Cabrera, waiting for either an amount of data to be backed up or an elapsed amount of time before initiating a backup, whether from the local storage to the staging area or from the staging area to remote storage. EX2020 ¶ 219.

The proposed alteration of Combination 2 of the Petition, which argues to “use a file-manipulation event . . . as taught by Schneider, within the archive-backup system of Cabrera” (Petition at 64), is unsupportable.

It would not have been obvious to take a periodic backup process, as taught in Cabrera, and change the principle of operation thereof to trigger a backup on a file manipulation event. EX2020 ¶ 220. Making the alteration proposed in the Petition, if it were even possible, which Datanet disputes, as discussed below, would completely change Cabrera. It would eliminate, for example, steps 124 and 126 from Cabrera, in which Cabrera checks for sufficient data before initiating backup from local storage. EX2020 ¶ 221. As taught by Cabrera, “decision block 124 and time delay 126 illustrate a mechanism whereby a system periodically checks to see if sufficient data resides in local storage to be staged to the staging. EX1005-10:45-49. The other alternative actually spelled out in Cabrera is “the triggering event for staging data to the staging area would be the expiration of an elapsed time rather than the accumulation of an amount of data.” EX1005-10:53-55. The backup is fundamentally a periodic process in Cabrera, and triggering backup based on a file manipulation event, as proposed in Ground 2, would not have been obvious. EX2020 ¶ 222.

b) The Proposed Combination Would Not Work in the System of Cabrera

A core premise of the Petition’s Ground 2 is that “Cabrera states that ‘any ... triggering event’ may be used to trigger copying data from local storage to the staging area” Petition 64 (quoting EX1005-10:56-61). The Institution Decision relies on this same section of Cabrera to show Limitation 1.2, with emphasis on Cabrera’s statement that “any other triggering event” can be used to trigger copying of data to Cabrera’s staging area. Institution Decision 34.

Ground 2’s combination purports to use Schneider’s teaching that, “when ‘a request to write [to the disk] occurs, . . . intercepting the write’ to trigger copying the file on the disk before completing the write to the disk.” Petition 67 (citing EX1020-12:5-8). Ground 2 uses this “intercepting the write” and essentially plugs this Schneider event into Cabrera’s “any other triggering event” to yield Limitation 1.2 of the challenged claims.

But, despite the broad wording of Cabrera’s fourth-listed potential trigger – “any other trigger[]” – it could not, in fact be any trigger under the sun. The proposal of using Schneider’s “intercepting the write” to trigger copying the file before completing the write (Petition 67) would not work as the triggering event for copying data from local storage to the staging area of Cabrera. EX2020 ¶ 225. As shown below, Cabrera’s backup operation is performed on data only *after* the data is present in the local storage area.

Under its description of the alleged meeting of Limitation 1.2 for Ground 2, the Petition specifically points to Cabrera at column 6, lines 58-62 as creating the “data that is to be backed up or archived.” Petition 67. This data to be backed up is the only aspect of Cabrera (or Schneider), that the Petition points to that could arguably be mapped to the recited “operating file.” Similarly, under Limitation 1.3, the Petition points to Cabrera’s disclosure that of “copying a file ‘from local storage to the staging file’ in response to ‘any . . . triggering event’” (Petition 68 (citing EX1005-10:56-61)) as meeting Limitation 1.3 for “creating an archive file from the operating file.” In short, the alleged “operating file” of Ground 2 is Cabrera’s data file (64) as shown in Figure 2 of Cabrera and its associated discussion in the Cabrera specification. EX2020 ¶ 226.

However, Cabrera’s data file is *already created / written to* before Cabrera’s backup procedure is initiated. That is how Cabrera works. As stated by Cabrera,

In FIG.2, one or more data producers, such as data producer 60, creates data that is to be backed up or archived to a backup or archive storage device, such as remote Storage 62. The data produced by data producer 60 is stored in a local storage medium, such as the hard disk for the local computer system. In FIG. 2, data producer 60 is illustrated as storing data in data file 64.

EX1005-6:58-67. This is the method that Cabrera uses to create data, and no other data creation method is disclosed or suggested by Cabrera. EX2020 ¶ 226.

Cabrera discusses moving its data files to the staging area as follows:

When a first event occurs, indicating that data local storage area should be transferred to a staging area, archive system 66 will move an appropriate amount of data from the local storage area, such as data file 64, to a staging storage area which is adapted for temporarily storing the data.

EX1005-7:3-7. Cabrera goes on to state that “the present invention may comprise means for moving data from a local storage area used for data storage by a data producing service to a staging storage area used for temporarily staging the data.”

EX1005-7:7-11. For example, copying data or direct memory transfer may be used to move the data to the staging area. EX1005-7:11-20.

This is the data that is backed-up by the Cabrera system, data produced by the data producing service, i.e., data *after* it is produced. Even though Cabrera uses the words “any triggering event,” Cabrera can only copy data that has already been written to the local disk. A POSITA would recognize that “intercepting the write to trigger copying the file on the disk *before completing the write*” (Petition 67) as proposed by the Petition *would not work in the system of Cabrera*, which copies or transfers data to the staging file only *after* it has been produced by the data producer. EX2020 ¶ 228. This cannot be done in Cabrera if the alleged “trigger” is before the write is even executed.

This is further illustrated through Figure 2 of Cabrera:

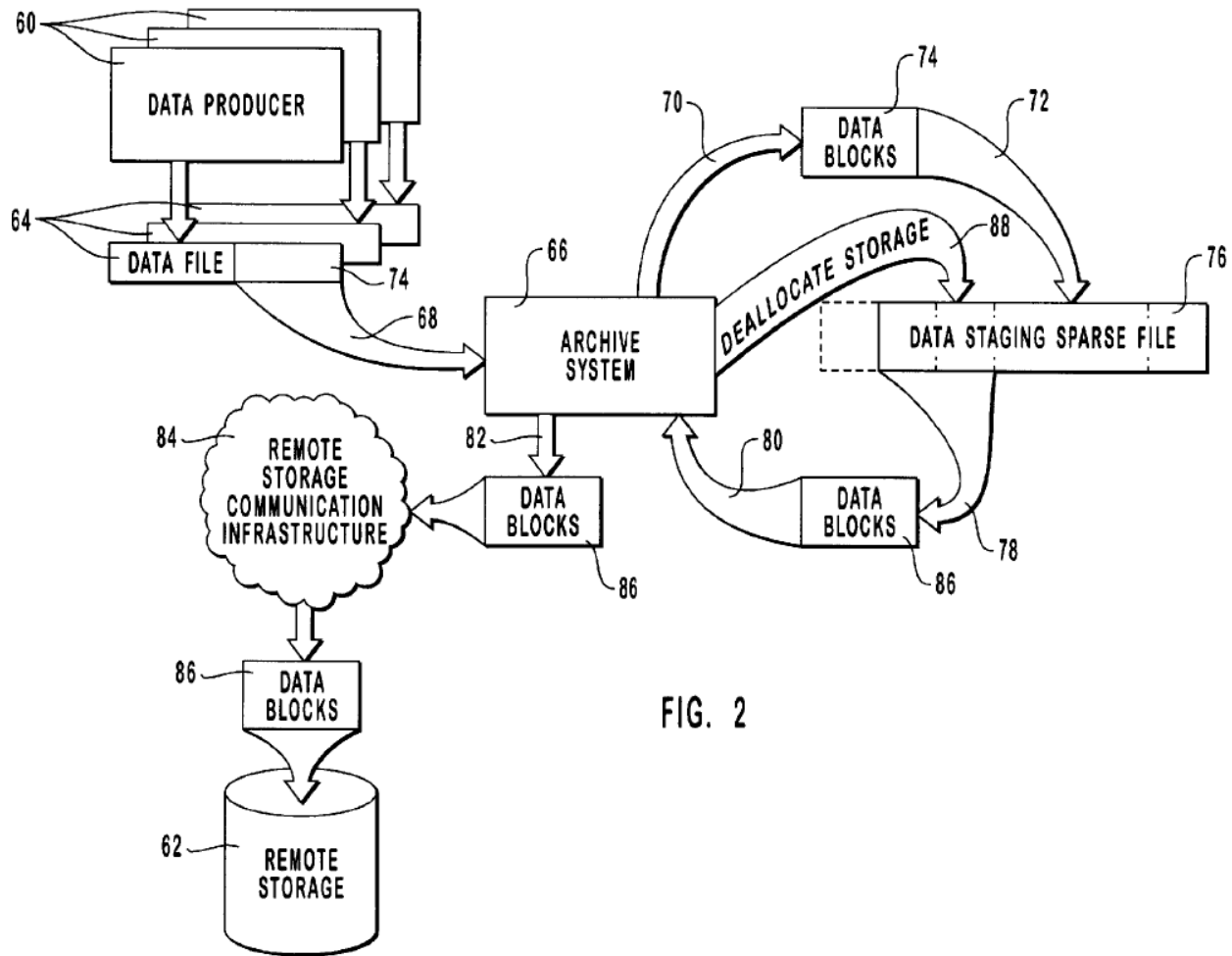


FIG. 2

EX1005-Fig. 2. As can be seen in Figure 2, Data Producer 60 produces Data File 64. These are the downward moving arrows from Data Producer 60 that produce the data in Data File 64. The arrow, 68 is the “means for moving data” from to the archive system. EX1005-7:16-20.

The data moving means, 68 comes *after* the arrows from the data producer to the data file, not before. EX2020 ¶ 229. This illustrates that the archive system of Cabrera is not interposed between the data producer and the data file. In Cabrera,

only data that has already been produced is backed up to the staging area. “Any triggering event” in Cabrera cannot be “intercepting the write” to trigger copying the file before completing the write (Petition 67) as proposed in the Petition, as it simply would not work in Cabrera. EX2020 ¶ 229.

* * *

Therefore, it would not have been obvious to combine Cabrera and Schneider as proposed in Ground 2 of the Petition because using Cabrera in the manner proposed would change the principle of operation of Cabrera, stripping out its core periodic nature. Further, combining Cabrera and Schneider as proposed, fundamentally, would not work, and as such a POSITA would have no reasonable expectation of success. As a result, Ground 2 fails as to all challenged claims.

7. Whiting Does Not Disclose or Suggest Updating a Database to Reflect Storage of an Archive File in Multiple Locations, and a POSITA Would Not Combine Whiting with Schneider and Cabrera

Whiting is used by the combination in Ground 2 to address the database limitations of the challenged claims, Limitations 1.6, 1.9, and 1.10.²

² Patent Owner notes that the inclusion of Whiting for the database limitations basically admits that the combination of Schneider and Cabrera lacks these limitations.

The Petition's overview discussion of Whiting does little to explain Whiting and its backup directory. Petition 62-63. It essentially argues that Whiting is a backup system with a directory. The backup directory is specific to each user. Petition 62-63.³

The Petition's discussion of Whiting is brief, and the Petition casually cites to many columns of Whiting without any accompanying discussion in the Petition itself. *See* Petition 63 (citing to EX1006 "8:41-13:9" and "20:6-26:36" without discussion thereof).

In arguing a motivation to combine, the Petition does not specify how Whiting would fit into the combination, just arguing to use the database of Whiting to improve Cabrera's alleged ability to track data. Petition 65-66.

The Petition essentially throws Whiting into the combination of Ground 2 with minimal discussion. Whiting has a backup directory file that allegedly stores data on the location of the files backed up by Whiting. There is no real discussion of Whiting in the Petition, or of how or why Whiting would fit into the combination with Schneider and Cabrera. The Petition just says it would be obvious to use

³ The global directory database of Whiting, also cited by the Petition, is used in Whiting merely to avoid making multiple copies of the same files. *See, e.g.*, EX1006-8:6-20.

Whiting's backup directory file, merely to check off the database feature of Limitations 1.6, 1.9, and 1.10.

Fundamentally, Whiting is *not* a multi-location backup system. EX2020 ¶ 233. All of the backup data in Whiting is stored in a single backup disk. *See, e.g.*, EX1006-Fig 1, 7:13-17 (“As shown in FIG. 2[,], each node is assigned two directories, a user directory and a system directory, on the backup storage means 101, which is contained in the disk volumes of the network file server 100.”). There is one network file server 100 with one backup storage means 101 in Whiting. Whiting does not disclose or suggest moving backup data between multiple backup locations, let alone updating any database to indicate that any archive file is located in each of the locations. EX2020 ¶ 233.

The Petition and the accompanying declaration merely point to the backup directory of Whiting as storing backup data locations. From this, the argument proceeds that it would be obvious to use such a system in a multi-location backup system like Cabrera and, in so doing, to track the location of the backed-up data as it is moved between the backup locations of Cabrera.

This combination and argument are pure hindsight, using the limitations of the challenged claims as a roadmap to find and combine prior art.

There is only a single paragraph of Whiting discussing that, in one embodiment, there could be hierarchical storage management (HSM), in which “files that have

not been accessed for a long time are migrated from disk to a secondary storage means, such as tape or optical disk,” which is done to save on storage costs. *See* EX1006-6:52-7:6. Whiting discloses nothing about how this HSM would work or how it would affect Whiting’s backup directory. EX2020 ¶ 234. The Petition is wholly unclear as to whether and how Whiting would track locations of data when it is stored in a secondary storage means.

In the discussion of Limitation 1.6, there is only one quote from Whiting: “Whiting discloses a ‘backup directory file’ that ‘indicate[s] the directory structure of the source disk volume, as well as pointers into the backup data files . . . indicating where the data for each file is to be found.’” Petition 71-72 (citing EX1006-8:45-50; EX1003 ¶ 167).

The Petition’s discussion for Limitations 1.9 and 1.10 simply point to the discussion for Limitation 1.6. Petition 74. This is because, contrary to Limitations 1.9 and 1.10, which recite updating the database to indicate that the archive file is in various storage locations, Whiting does not have multiple backup data storage locations and does not track the movement of backup data between such locations. EX2020 ¶ 236.

For these additional reasons, (1) the failure of Whiting to track locations of its data in various backup locations, and (2) the hindsight used to even attempt adding Whiting to the combination of Ground 2, Ground 2 fails to disclose or suggest

Limitations 1.6, 1.9, and 1.10 and therefore fails to render claim 1 obvious. For at least the same reasons, Ground 2 fails to disclose or suggest Limitations 8.6, 8.8, which the Petition maps to the same disclosure as 1.9 (Petition at 47), 8.10, which the Petition maps to the same disclosure as 1.10 (Petition at 48), and therefore fails as to claim 8. Dependent claims 2-6 depend from claim 1 and dependent claims 9-11 depend from claim 8 and are patentable for at least the same reasons. EX2020 ¶ 237.

VI. CONCLUSION

For the foregoing reasons, the Board should find all challenged claims patentable.

Dated: August 2, 2024

Respectfully submitted,

/Ari J. Jaffess/

Ari J. Jaffess

(Attorney for Patent Owner)

Reg. No. 74,558

212-257-1630

CERTIFICATION UNDER 37 CFR § 42.24(d)

Pursuant to 37 CFR §42.24(d), the undersigned hereby certifies that the word count for the foregoing Patent Owner Preliminary Response totals 12,939 words, which is less than the 14,000 allowed under 37 CFR §42.24(a)(1)(i).

Dated: August 2, 2024

Respectfully submitted,

/Ari J. Jaffess/

Ari J. Jaffess

(Attorney for Patent Owner)

Reg. No. 74,558

212-257-1630

CERTIFICATE OF SERVICE

Pursuant to 37 CFR § 42.205(b), the undersigned certifies that on August 2, 2024, a complete and entire copy of this Patent Owner Response was served on Petitioners by filing through the Patent Trial and Appeal Case Tracking System (P-TACTS) and via email to andrew.mason@klarquist.com, cameron.clawson@klarquist.com, sarah.jelsema@klarquist.com, and frank.morton-park@klarquist.com, as authorized in Petitioners' mandatory notices.

Dated: August 2, 2024

Respectfully submitted,

/Ari J. Jaffess/

Ari J. Jaffess

(Attorney for Patent Owner)

Reg. No. 74,558

212-257-1630