

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

MICROSOFT CORP.,
Petitioner,

v.

ROYAL J. O'BRIEN,
Patent Owner.

IPR2021-00015
Patent 8,380,808 B2

Before GEORGIANNA W. BRADEN, KEVIN C. TROCK, and
STEVEN M. AMUNDSON, *Administrative Patent Judges*.

AMUNDSON, *Administrative Patent Judge*.

JUDGMENT
Final Written Decision
Determining All Challenged Claims Unpatentable
Denying Petitioner's Motion to Exclude and Strike
35 U.S.C. § 318(a)

I. INTRODUCTION

Microsoft Corp. (“Petitioner”) filed a Petition requesting an *inter partes* review of claims 1–20 in U.S. Patent No. 8,380,808 B2 (Exhibit 1001, “the ’808 patent”) under 35 U.S.C. §§ 311–319. Paper 1 (“Pet.”). Royal J. O’Brien (“Patent Owner”) filed a Preliminary Response. Paper 6 (“Prelim. Resp.”).

In our Institution Decision, we instituted review based on all challenged claims and all grounds presented in the Petition. Paper 8 (“Inst. Dec.”). We have jurisdiction under 35 U.S.C. § 6. We issue this Final Written Decision under 35 U.S.C. § 318(a) and 37 C.F.R. § 42.73. For the reasons explained below, Petitioner has shown by a preponderance of the evidence that claims 1–20 in the ’808 patent are unpatentable. *See* 35 U.S.C. § 316(e) (2018).

II. BACKGROUND

A. Procedural History

After we instituted review, Patent Owner filed a Response (Paper 12, “Resp.”)¹, Petitioner filed a Reply (Paper 14, “Reply”), and Patent Owner filed a Sur-reply (Paper 16, “Sur-reply”). On January 28, 2022, we held an oral hearing, and the record includes the hearing transcript. Paper 31 (“Tr.”).

On January 31, 2022, we issued an order authorizing the parties to file opening and response briefs addressing the patentability of claim 13 in the ’808 patent assuming that (1) the term “extended data” means “data related

¹ Patent Owner filed a Response (Paper 10) on July 22, 2021, and a corrected Response (Paper 12) on August 9, 2021. We reference the corrected Response.

to a file,” such as a file name or address, and (2) the term “file stream” means “data structure for file data,” such as a FILE_OBJECT data structure. Paper 26. Then, Patent Owner filed an opening brief (Paper 27), Petitioner filed an opening brief (Paper 28), Petitioner filed a response brief (Paper 29), and Patent Owner filed a response brief (Paper 30).

B. Real Parties in Interest

Petitioner identifies itself as the real party in interest. Pet. 3. Patent Owner identifies himself as the real party in interest. Paper 5, 1. The parties do not raise any issue about real parties in interest.

C. Related Matters

Petitioner and Patent Owner identify the following civil action where Patent Owner has asserted the '808 patent against Petitioner: *Royal J. O'Brien v. Microsoft Corp.*, 2:19-cv-01625-RAJ (W.D. Wash. filed October 10, 2019). Pet. 4; Paper 5, 1.

D. The '808 Patent (Exhibit 1001)

The '808 patent, titled “Dynamic Medium Content Streaming System,” issued on February 19, 2013, from an application filed on November 24, 2008. Ex. 1001, codes (22), (45), (54). The patent states that the invention concerns “streaming software” and managing “delivery of software from a delivery source” in “separate executable portions.” *Id.* at 1:4–9; *see id.* at 2:1–3, code (57).

The '808 patent explains that “streaming software” involves “preprocessing that entails separating a program into small manageable packages that are stored on a server.” Ex. 1001, 1:34–36. The “packages are downloaded as needed by the program being streamed.” *Id.* at 1:36–38.

Hence, “only those portions of the program or data actually used are downloaded.” *Id.* at 1:44–46.

For example, a program’s files are “broken into compressed, encrypted chunks in a preprocessing phase.” Ex. 1001, 1:41–47 (citing U.S. Patent No. 6,453,334 B1 (Ex. 2067)). The preprocessing phase includes the following steps:

- (1) “opening the file”;
- (2) “reading a block of chunk-size bytes”;
- (3) “compressing the block”;
- (4) “encrypting the block”;
- (5) “calculating a unique file name for the block from the unique ID of the file and the block number”;
- (6) writing the processed block to a new file in a specified directory in a server using the file name just generated; and
- (7) “repeating for the next block in the input file until the end of file is reached.”

Id. at 1:47–54.

Thus, the preprocessing phase produces chunks “formed sequentially, one after another, comprising contiguous portions of the executable program.” Ex. 1001, 1:55–56. After the preprocessing phase, an “index contains the information used by a client-side file system driver (FSD) to compute which chunk to download at runtime as it is needed.” *Id.* at 1:56–59.

According to the ’808 patent, “[t]here are many problems associated with streaming software that it would be advantageous to negate, work around, or reduce.” Ex. 1001, 2:4–6. As an example, the patent describes

“operating systems and security software” that “limit, impede or reject virtual drives and file system drivers” for security reasons as a “key problem.” *Id.* at 2:15–18; *see id.* at 10:65–11:2. The patent identifies a need for “a system and method” that permit “streaming delivery of program components” from a source “without creating a virtual drive or file system driver.” *Id.* at 2:32–35; *see id.* at 10:62–65, 11:2–19, 12:16–20.

The '808 patent's Figure 2 is reproduced below:

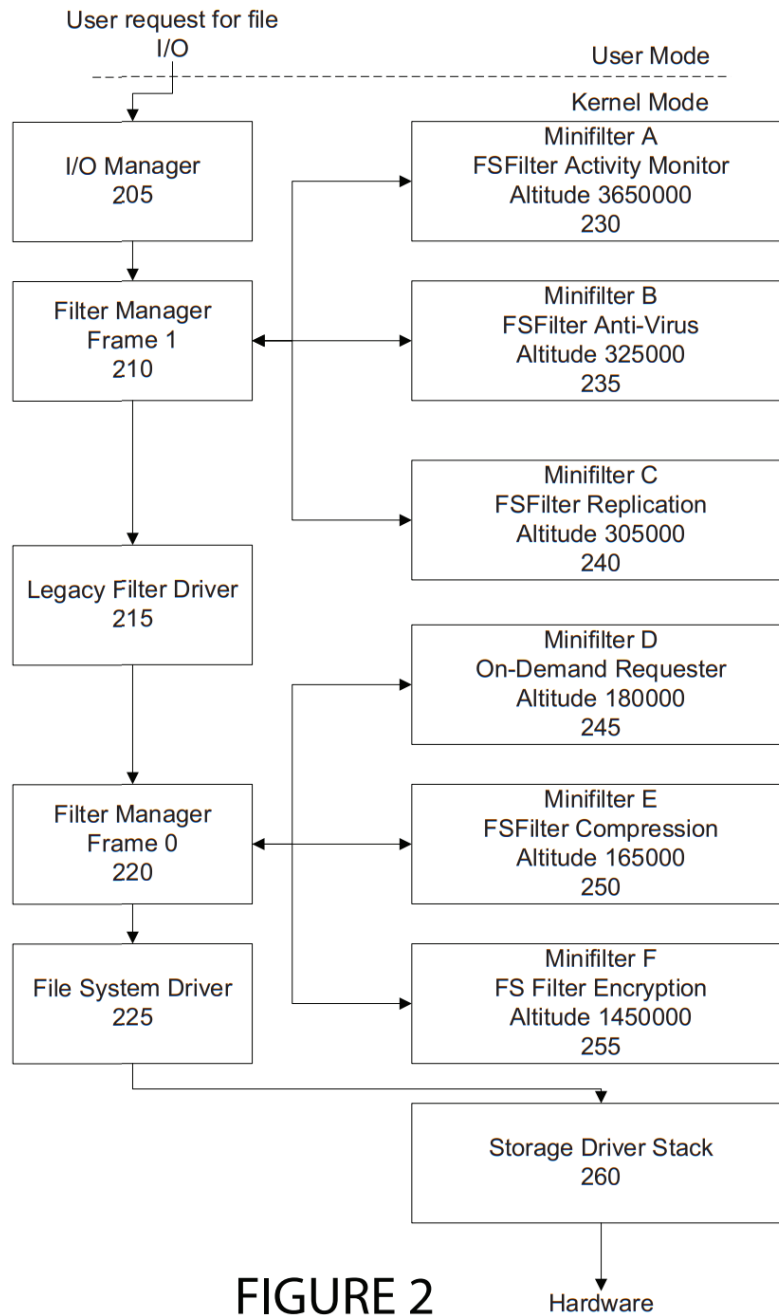


FIGURE 2

Figure 2, above, is a “flowchart of filter management aspects of an exemplary dynamic medium content streaming method.” Ex. 1001, 3:51–53, Fig. 2.

As Figure 2 shows, an input/output (I/O) stack includes I/O manager 205, first filter manager 210, legacy filter driver 215, second filter manager 220, and file system driver 225. Ex. 1001, 5:3–5, 5:17–26, Fig. 2. Minifilters 230, 235, and 240 attach to filter manager 210. *Id.* at 5:18–19. Minifilters 245, 250, and 255 attach to filter manager 220. *Id.* at 5:19–21. A minifilter “attaches to the file system stack indirectly” by “registering with the filter manager” for “the I/O operations the minifilter driver chooses to filter.” *Id.* at 5:29–32; *see id.* at 2:55–58, 3:26–29, 6:4–8.

Figure 2 describes minifilters 230–255 as follows:

- minifilter 230: Minifilter A FSFilter Activity Monitor;
- minifilter 235: Minifilter B FSFilter Anti-Virus;
- minifilter 240: Minifilter C FSFilter Replication;
- minifilter 245: Minifilter D On-Demand Requester;
- minifilter 250: Minifilter E FSFilter Compression; and
- minifilter 255: Minifilter F FSFilter Encryption.

Ex. 1001, Fig. 2.

The '808 patent explains that configuring a filter manager to communicate with “a managed filter or minifilter” that registers for particular I/O operations “simplifies processing I/O operations” compared to a legacy filter. Ex. 1001, 5:45–51; *see id.* at 4:11–16. Specifically, a legacy filter “pass[es] all I/O requests to the next-lower driver,” whereas a minifilter “registers only for the I/O operations it must handle.” *Id.* at 5:46–51. Thus, a minifilter “does whatever processing is needed for the I/O operation” and returns to the attached filter manager “the appropriate value from” a callback routine. *Id.* at 5:54–57. A filter manager may “create a data structure” for the callback data. *Id.* at 6:1–4.

Minifilters attach to a filter manager in a particular order determined by a unique identifier called an “altitude.” Ex. 1001, 5:32–34; *see id.* at 6:10–13. The altitude “ensures that the instance of the minifilter driver is always loaded at the appropriate location relative to” other minifilters. *Id.* at 5:34–36. For example, Figure 2 depicts on-demand requester minifilter 245 loading before compression minifilter 250 and encryption minifilter 255 “to avoid the complexity of decompression and decryption.” *Id.* at 5:39–42, Fig. 2.

The ’808 patent describes on-demand requester minifilter 245 as “unique to the subject invention.” Ex. 1001, 5:20–22; *see id.* at 5:13–16. Among other things, on-demand requester minifilter 245 “watches for any requests made by” a program identified in a watch list. *Id.* at 8:19–26. For a program identified in the watch list, on-demand requester minifilter 245 processes a read request by referencing a lookup table to determine if all the required data has been received and is “available locally.” *Id.* at 8:44–51; *see id.* at 2:48–51, 6:52–54, 6:62–63, code (57), Fig. 3A (steps 304, 306, and 308). If all the required data has been received and is “available locally,” on-demand requester minifilter 245 passes the read request “on to the next filter and processing continues in a normal manner.” *Id.* at 8:48–50; *see id.* at 2:59–61, 3:29–31, 6:54–56, code (57), Fig. 3A (step 310). But if all the required data has not been received or is not “available locally,” on-demand requester minifilter 245 withholds the read request from the next filter and sends “a new request to retrieve the data” to the user mode. *Id.* at 8:52–54; *see id.* at 2:61–63, 3:31–34, 6:66–67, code (57), Fig. 3A (step 320).

The lookup table includes addresses for the locations of the data needed for program operation. *See* Ex. 1001, 2:64–3:3, 6:44–49, 6:62–63, 7:1–5, code (57). The lookup table may include addresses at the following locations:

- (1) a “local source,” such as “a hard disk, an optical storage medium (e.g., [compact disc] CD or [digital versatile disc] DVD), a nonvolatile memory (e.g., a flash [random-access memory] RAM), or a magnetic storage medium (e.g., a diskette)”;
- (2) a “remote source,” such as “a server (e.g., [file transfer protocol] ftp and/or web server) and a peer computing device.”

Id. at 2:64–3:3; *see id.* at 7:5–9, 8:67–9:12. The lookup table may also include a “size indicator,” such as “start and end offsets,” for the data. *Id.* at 6:44–47; *see id.* at 2:52–53, code (57).

“Attempts to service each request may be made at any accessible source, such as a web server, [file transfer protocol] FTP, [peer to peer] P2P, hard disk, CD/DVD, flash memory, or any other form of medium that can store data.” Ex. 1001, 8:67–9:3; *see id.* at 3:3–7, 3:35–39, 7:5–9. “If only part of the data is found at one location, control will pass from location to location to retrieve all pieces of the data.” *Id.* at 9:3–5; *see id.* at 2:64–3:7, 7:5–9, Fig. 3B.

The '808 patent's Figure 4 is reproduced below:

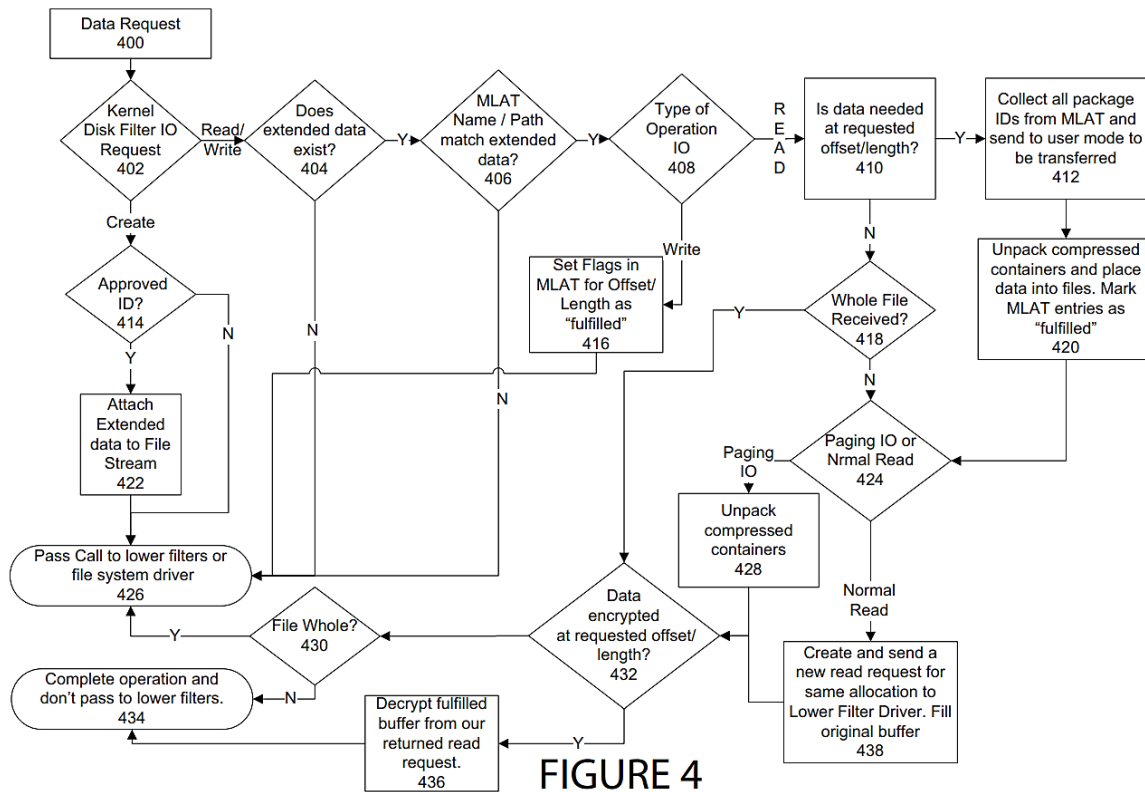


Figure 4, above, is a “flowchart of an exemplary method of processing data requests for a dynamic medium content streaming method.” Ex. 1001, 3:61–64, Fig. 4.

As Figure 4 shows, the method initially determines the type of “application data request,” e.g., a read, write, or create operation. Ex. 1001, 7:23–32, 8:12–13, Fig. 4. The discussion above about on-demand requester minifilter 245 summarizes the processing for a read request. *See id.* at 2:59–64, 3:29–38, code (57), Fig. 4. For “a write operation, the minifilter writes an address and size indicator corresponding to the write operation” to the lookup table, thus indicating to a later read request that the data has already been delivered. *Id.* at 3:8–11, code (57), Fig. 4; *see id.* at 7:61–65. For “a create operation, the minifilter attaches extended data to a file stream.” *Id.* at 3:11–13, code (57), Fig. 4; *see id.* at 8:12–16.

E. The Challenged Claims

Petitioner challenges independent claim 1, claims 2–13 that depend directly from claim 1, independent claim 14, and claims 15–20 that depend directly from claim 14. Claim 1 exemplifies the challenged claims and reads as follows (with formatting added for clarity and with letters and numbers added for reference purposes²):

1. [1.a] A streaming on demand system comprising
 - [1.b.i] an on-demand requester object installed on a computing device,
 - [1.b.ii] said on-demand requester object being configured to receive I/O requests on behalf of an application for which data is available in data packs for streaming delivery,
 - [1.b.iii] said on-demand requester object being responsive to application I/O requests,
 - [1.b.iv] said application I/O requests corresponding to sequential and non-sequential data packs,
 - [1.b.v] said data packs being portions of a complete data stream,
 - [1.b.vi] said I/O requests determining an order in which data packs are sought for the application,
 - [1.c.i] said on-demand requester object comprising a minifilter associated with a filter manager in an I/O stack,
 - [1.c.ii] said filter manager being configured to call the minifilter,
 - [1.c.iii] said minifilter having a preoperation callback routine for the I/O request registered with the filter manager, and

² We use the same letters and numbers that Petitioner uses to identify claim 1's limitations, but this formatting does not impact our analysis.

[1.c.iv] said minifilter being configured to receive each I/O request from the application,

[1.c.v] said minifilter being further configured to reference a table that includes names and paths with at least one offset address and length where each data pack required to fulfill each I/O request is located,

[1.c.vi] said minifilter being further configured to determine if the data pack has been streamed to the system, and

[1.d] said application being a program configured to use the data packs that fulfill each I/O request when the data packs are available on the computing device.

Ex. 1001, 12:44–67.

F. The Asserted References of Record

For its challenges, Petitioner relies on the following references:

Name	Reference	Exhibit
Eylon	US 6,574,618, issued June 3, 2003	1005
Christiansen	US 2006/0117018, published June 1, 2006	1006
Pudipeddi	US 6,993,603, issued January 31, 2006	1007
Thind	US 7,496,565, issued February 4, 2009	1008
Prahlad	US 7,617,253, issued November 10, 2009	1016

Petitioner asserts that Eylon, Christiansen, Pudipeddi, Thind, and Prahlad qualify as prior art under 35 U.S.C. §§ 102(a), (b), and (e).³ Pet. 19, 22, 65, 68, 72. For Thind and Prahlad, Petitioner bases the assertion about § 102(b) on the patent application’s publication date, i.e., a June 1, 2006,

³ The Leahy-Smith America Invents Act (“AIA”), Pub. L. No. 112-29, 125 Stat. 284 (2011), amended 35 U.S.C. § 102 and § 103 effective March 16, 2013. Because the ’808 patent’s filing date predates the AIA’s amendments to § 102 and § 103, this decision refers to the pre-AIA versions of § 102 and § 103.

publication date for the Thind application and an August 9, 2007, publication date for the Prahlad application. *Id.* at 68, 72. Patent Owner does not dispute that each reference qualifies as prior art. *See, e.g.,* Resp. 15–48.

G. The Asserted Challenges to Patentability

Petitioner asserts the following challenges to patentability:

Claim(s) Challenged	35 U.S.C. §	Reference(s)/Basis
1–20	103(a)	Christiansen, Eylon
2	103(a)	Christiansen, Eylon, Pudipeddi
12	103(a)	Christiansen, Eylon, Prahlad
13	103(a)	Christiansen, Eylon, Thind

Pet. 5, 30–74.

H. Testimonial Evidence

To support its challenges, Petitioner relies on the following declarations: (1) Declaration of Henry Houh Regarding U.S. Patent No. 8,380,808 (Ex. 1003, “Houh 1st Decl.”); (2) Reply Declaration of Henry Houh Regarding U.S. Patent No. 8,380,808 (Ex. 1035, “Houh 2nd Decl.”); and (3) Declaration of Brian Catlin Regarding U.S. Patent No. 8,380,808 (Ex. 1026, “Catlin Decl.”). Dr. Houh states, “I have been retained by counsel for Petitioner” as “an expert witness” and “have been asked to provide my opinion about the state of the art of the technology described” in the ’808 patent and “on the patentability of claims 1–20.” Ex. 1003 ¶ 2. Mr. Catlin states, “I have been retained by counsel for Petitioner” as “an expert witness” and “have been asked to provide my opinion about the state of the art of the technology described” in the ’808 patent “in the 2008 time frame.” Ex. 1026 ¶ 2.

To support his positions, Patent Owner relies on the following declarations: (1) Declaration of William Anthony Mason (Paper 7, “Mason 1st Decl.”); (2) Declaration of William Anthony Mason (Ex. 2043, “Mason 2nd Decl.”); and (3) Declaration of Royal O’Brien (Ex. 2105, “O’Brien Decl.”). Mr. Mason states, “I have been retained as an independent expert on behalf” Patent Owner to “provide my analyses and opinions on certain technical issues related to” the ’808 patent. Paper 7 ¶ 1. Mr. O’Brien states, “I am the named inventor on” the ’808 patent. Ex. 2105 § 1 ¶ 1.

Additionally, Patent Owner submits deposition testimony from Dr. Houh. *See* Ex. 2134 (Nov. 23, 2021, Houh Dep. Tr.).

I. Burden

In an *inter partes* review, the petitioner bears the burden of persuasion to prove “unpatentability by a preponderance of the evidence.” *Dynamic Drinkware, LLC v. Nat’l Graphics, Inc.*, 800 F.3d 1375, 1378 (Fed. Cir. 2015) (quoting 35 U.S.C. § 316(e)); *see* 37 C.F.R. § 42.1(d) (2021).

III. PATENTABILITY ANALYSIS

A. Legal Principles: Obviousness

A patent may not be obtained “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.” 35 U.S.C. § 103(a) (2006). An obviousness analysis involves underlying factual inquiries including (1) the scope and content of the prior art; (2) differences between the claimed invention and the prior art; (3) the level of ordinary skill in the art; and (4) where in evidence, objective indicia of nonobviousness, such as commercial success, long-felt but unmet need, and

failure of others. *Graham v. John Deere Co.*, 383 U.S. 1, 17–18, 35–36 (1966); *Apple Inc. v. Samsung Elecs. Co.*, 839 F.3d 1034, 1047–48 (Fed. Cir. 2016) (en banc). When evaluating a combination of references, an obviousness analysis should address “whether there was an apparent reason to combine the known elements in the fashion claimed by the patent at issue.” *KSR Int’l Co. v. Teleflex Inc.*, 550 U.S. 398, 418 (2007).

We analyze the obviousness issues according to these principles.

B. Level of Ordinary Skill in the Art

Factors pertinent to determining the level of ordinary skill in the art include (1) the educational level of the inventor; (2) the type of problems encountered in the art; (3) prior-art solutions to those problems; (4) the rapidity with which innovations are made; (5) the sophistication of the technology; and (6) the educational level of workers active in the field. *Envtl. Designs, Ltd. v. Union Oil Co.*, 713 F.2d 693, 696–97 (Fed. Cir. 1983). Not all factors may exist in every case, and one or more of these or other factors may predominate in a particular case. *Id.* These factors are not exhaustive, but merely a guide to determining the level of ordinary skill in the art. *Daiichi Sankyo Co. v. Apotex, Inc.*, 501 F.3d 1254, 1256 (Fed. Cir. 2007). Moreover, the prior art itself may reflect an appropriate skill level. *Okajima v. Bourdeau*, 261 F.3d 1350, 1355 (Fed. Cir. 2001).

Based on testimony from Dr. Houh and Mr. Catlin, Petitioner asserts that a person of ordinary skill in the art would have had the following qualifications:

- (1) “at least a bachelor’s degree in computer science, or a related field”;
- (2) “four years of relevant work experience, such as writing software for streaming applications”; and

- (3) “familiar[ity] with known technologies, such as streaming technology, operating system methods for handling I/O requests, and loadable device driver management architectures such as filter manager and minifilter architecture.”

Pet. 6 (citing Ex. 1003 ¶ 47; Ex. 1026 ¶ 11). Petitioner asserts that “[a]lternatively, such a person would have had at least a master’s degree in computer science, or a related field, with two years of work experience.” *Id.* According to Petitioner, a “lack of work experience can be substituted by additional education, and vice versa.” *Id.*

In the Response, Patent Owner does not propose an educational level or an amount of work experience for an ordinarily skilled artisan. *See* Resp. 8–11. Rather, Patent Owner criticizes Petitioner’s description of an ordinarily skilled artisan for including familiarity with “streaming technology, operating system methods for handling I/O requests, and loadable device driver management architectures.” *See id.* at 9. Patent Owner asserts that “[e]ach of these are highly specialized fields” that “do not necessarily overlap in any fashion” and “are not taught in the standard curriculum for undergraduate studies.” *Id.* Patent Owner also asserts that “[t]here is no curriculum that could educate the expert as suggested.” *Id.* at 9–10 (citing Exs. 2043, 2068–2080).

In the Sur-reply, Patent Owner asserts that an ordinarily skilled artisan would have required the following qualifications to combine Eylon’s teachings with Christiansen’s teachings as Petitioner proposes:

- (1) “a background in network data streaming technologies; caching mechanisms; streaming game delivery; streaming application delivery; user mode programming, and kernel mode programming,” including “file systems,

file system filter drivers, and file system mini-filter drivers”;

- (2) “deep knowledge” of “compression and encryption filters”;
- (3) “knowledge” of “stream handle context” and “file context” as “these are fundamental concepts of context management”; and
- (4) “insight to understand combining a server side component” with “client side components.”

Sur-reply 2.

At the oral hearing, Patent Owner described an ordinarily skilled artisan as someone with either a “Master’s level education” or “at least four years of experience in filter architecture.” Tr. 28:10–29:5.

We adopt Petitioner’s description of an ordinarily skilled artisan because it comports with the technology and claims in the ’808 patent as well as the asserted prior art. Petitioner and Patent Owner agree about an ordinarily skilled artisan’s educational level, i.e., Master’s degree, but disagree about the amount of work experience, i.e., two years according to Petitioner and four years according to Patent Owner. *See* Pet. 6; Tr. 28:10–29:5. If the prior art renders the claimed subject matter obvious to a person with less experience and less knowledge, then the prior art renders the claimed subject matter obvious to a person with more experience and more knowledge. *See Tokai Corp. v. Easton Enters., Inc.*, 632 F.3d 1358, 1369 (Fed. Cir. 2011). We note that our analysis would not change if we adopted Patent Owner’s description of an ordinarily skilled artisan.

As for Patent Owner’s criticism that Petitioner’s description wrongly includes familiarity with certain topics that allegedly “are not taught in the standard curriculum for undergraduate studies,” an ordinarily skilled artisan

likely would have acquired familiarity with those topics through work experience or post-degree seminars (or both). *See* Ex. 1003 ¶ 47; Ex. 1026 ¶ 11; Ex. 2134, 26:19–28:4, 28:15–29:3; Paper 7 ¶¶ 4, 88.

Patent Owner asserts that Dr. Houh lacks the qualifications of an ordinarily skilled artisan. Sur-reply 2–3. We disagree. Dr. Houh has a Ph.D. in Electrical Engineering and Computer Science and more than four years of relevant work experience. *See* Ex. 1003 ¶¶ 4–11; Ex. 1004, 1–4. Moreover, he has familiarity with known technologies, such as streaming technology, operating system methods for handling I/O requests, and loadable device driver management architectures. *See, e.g.*, Ex. 1003 ¶¶ 7, 9, 11; Ex. 2134, 8:13–24, 9:4–10:20, 11:8–23, 13:3–21, 15:22–17:4, 17:22–18:13, 30:12–32:4, 109:24–110:17. Hence, Dr. Houh has at least ordinary skill in the art.

There is no requirement for a perfect match between an expert’s qualifications and the relevant field. *See SEB S.A. v. Montgomery Ward & Co.*, 594 F.3d 1360, 1373 (Fed. Cir. 2010); Consolidated Trial Practice Guide⁴ at 34. We have considered Dr. Houh’s qualifications in evaluating the weight given to his testimony.

C. Claim Construction

1. GENERALLY

Because Petitioner filed the Petition after November 13, 2018, we construe claim terms “using the same claim construction standard” that district courts use to construe claim terms in civil actions under 35 U.S.C. § 282(b). *See* 37 C.F.R. § 42.100(b). Under that standard, “[c]laim terms

⁴ Available at <https://www.uspto.gov/TrialPracticeGuideConsolidated>.

are given their ordinary and customary meaning, which is the meaning the term would have to a person of ordinary skill in the art at the time of the invention.” *Power Integrations, Inc. v. Fairchild Semiconductor Int’l, Inc.*, 904 F.3d 965, 971 (Fed. Cir. 2018) (citing *Phillips v. AWH Corp.*, 415 F.3d 1303, 1312–13 (Fed. Cir. 2005) (en banc)). The meaning of claim terms may be determined by “look[ing] principally to the intrinsic evidence of record, examining the claim language itself, the written description, and the prosecution history, if in evidence.” *DePuy Spine, Inc. v. Medtronic Sofamor Danek, Inc.*, 469 F.3d 1005, 1014 (Fed. Cir. 2006) (citing *Phillips*, 415 F.3d at 1312–17).

Petitioner proposes that we construe twelve terms or phrases in the ’808 patent’s claims. *See* Pet. 10–19; Reply 2–8. Patent Owner proposes that we construe eight terms or phrases. *See* Resp. 12–15. To decide the patentability issues for claims 1–20, we need to address the meaning of only five terms: “data packs,” “minifilter,” “filter minifilter driver,” “extended data,” and “file stream.”

2. “DATA PACKS”

Independent claims 1 and 14 recite “data packs for streaming delivery” and “said data packs being portions of a complete data stream.” Ex. 1001, 12:48–52, 14:3–8.

Petitioner proposes that we construe “data packs” to mean “portions of a data file that has been broken up for streaming.” Pet. 10 (citing Ex. 1003 ¶ 68). Specifically, Petitioner asserts that the ’808 patent lists “video, game or program data” as examples of “the data that is streamed in the disclosed system.” *Id.* at 11 (quoting Ex. 1001, 6:38). Petitioner also asserts that during prosecution the Applicant distinguished the application

claims from the Robert reference by arguing that “Robert accesses only files, not pieces of data such as the data packs referenced in Applicant’s Claim 1.” *Id.* (emphasis omitted) (quoting Ex. 1002, 81).

Patent Owner proposes that we construe “data packs” to mean “an asset consisting of some number of bytes of information” where “such information may be present in multiple locations across multiple files.” Resp. 14. Specifically, Patent Owner asserts that the ’808 patent “uses application I/O to identify the structure of the data within the file” and “then constructs a definition of those elements by defining each element (‘asset’ or ‘data pack’).” *Id.* To support that assertion, however, Patent Owner cites no part of the ’808 patent. *Id.*

For the reasons explained below (and in our Institution Decision), we construe “data packs” to mean “portions of a file that has been broken up for streaming.” *See* Inst. Dec. 16–23.

First, the claim language supports the conclusion that “data packs” means “portions of a file that has been broken up for streaming” because claims 1 and 14 require that “data packs” constitute “**portions** of a complete data stream.” Ex. 1001, 12:51–52, 14:7–8 (emphasis added).

Second, the ’808 patent’s specification supports the conclusion that “data packs” means “portions of a file that has been broken up for streaming” because the specification indicates that “a complete data stream” corresponds to a file. *See* Ex. 1001, 1:44–56, 6:62–65, 7:34–37, 8:30–55, 10:20–41, 11:31–58, Fig. 3A (step 318), Fig. 4 (step 420). Furthermore, the specification equates “data packs” to “assets.” *Id.* at 7:1, 7:19–20. For instance, the specification uses “data pack” and “asset pack” synonymously. *Id.* at 7:19–21, 8:4–9, Fig. 3B (step 344); *see* Paper 7 ¶¶ 37, 54, 58, 105,

112, 146–147. The specification describes “assets” as “files or file portions” that “are broken up” for streaming. Ex. 1001, 10:26–36.

The ’808 patent’s specification discloses breaking up files into portions for streaming. Ex. 1001, 10:20–27. In particular, the specification explains that “[a] streaming system according to principles of the invention relies upon a pre-structured package system that contains an organized object grouping” and that “[e]ach group may contain smaller portions of the original files.” *Id.* at 10:20–23. By breaking up “the original files” into portions for streaming, the portions “may be organized depending upon how and when the portions are used and the bandwidth consumed by the portion.” *Id.* at 10:23–25.

In addition, the ’808 patent’s specification includes an example of breaking up a file into portions for streaming. Ex. 1001, 8:29–9:5. In that example, “the whole file DATA.DAT” for a sound clip comprises 200 KB. *Id.* at 8:33–36. If all the required data for file DATA.DAT is not available locally, attempts to obtain the required data “may be made at any accessible source,” and “[i]f only part of the data is found at one location, control will pass from location to location to retrieve all pieces of the data.” *Id.* at 8:62–9:5. In that example, “the whole file DATA.DAT” has been broken up into portions for streaming.

Third, the ’808 patent’s prosecution history supports the conclusion that “data packs” means “portions of a file that has been broken up for streaming.” In particular, application claims 1 and 14 became patent claims 1 and 14, respectively. Ex. 1002, 117. In the first Office action, the Examiner rejected application claims 1 and 14 under § 103(a) as obvious

over Robert and Poli.⁵ *Id.* at 61–62. In doing so, the Examiner cited Robert as teaching “data packs for streaming delivery.” *Id.* at 61 (emphasis omitted) (citing Ex. 1019 ¶¶ 9–10, code (57)).

In response to the first Office action, the Applicant amended application claims 1 and 14 to require that “data packs” constitute “**portions** of a complete data stream,” such as a file. Ex. 1002, 77–79 (emphasis added). The Applicant distinguished the amended claims from the references by arguing that “Robert accesses only files, not pieces of data such as the data packs referenced in Applicant’s Claim 1.” *Id.* at 81. Thus, the Applicant took the position that “data packs” correspond to “pieces of data,” such as portions of a file.

Petitioner proposes a construction for “data packs” limited to “portions of a data file.” *See* Pet. 10–11. But the ’808 patent’s specification indicates that “data packs” encompass not only portions of data files but also portions of program files with executable code. *See* Ex. 1001, 1:4–9, 1:44–56, 2:1–3, 2:32–35, 9:8–23, code (57). For instance, the patent explains that the invention concerns “streaming software” and managing “delivery of software from a delivery source” in “separate executable portions.” *Id.* at 1:4–9; *see id.* at 2:1–3, code (57). The “separate executable portions” include program files with executable code.

⁵ *See* U.S. Patent Application Publication 2006/0101476, titled “Method and System for Recording and Replaying Input-Output Requests Issued by a User-Mode Program,” to Christophe F. Robert (“Robert”) (Ex. 1019); U.S. Patent Application Publication 2008/0075285, titled “Method and Apparatus for Delivering Encrypted On-Demand Content Without Use of an Application Defined Protocol,” to Christopher Poli et al. (“Poli”) (Ex. 1020).

The '808 patent also explains that “streaming software” involves “preprocessing that entails separating a program into small manageable packages that are stored on a server.” Ex. 1001, 1:34–36. The “packages are downloaded as needed by the program being streamed.” *Id.* at 1:36–38. Hence, “only those portions of the program or data actually used are downloaded.” *Id.* at 1:44–46. Thus, the patent contemplates streaming “portions of the program or data,” i.e., portions of program files and portions of data files.

Furthermore, the '808 patent describes “operating systems and security software” that “limit, impede or reject virtual drives and file system drivers” for security reasons as a “key problem.” Ex. 1001, 2:15–18; *see id.* at 10:65–11:2. The patent identifies a need for “a system and method” that permit “streaming delivery of program components” from a source “without creating a virtual drive or file system driver.” *Id.* at 2:32–35; *see id.* at 10:62–65, 11:2–19, 12:16–20. The “program components” include program files and data files.

In addition, the '808 patent discloses an example where a 5.0-GB game application has program components stored in various locations, i.e., 0.4 GB stored on a game CD, 1.6 GB stored on a webserver, and 3.0 GB stored on a peer-to-peer network. Ex. 1001, 9:8–23. The patent explains that “data for common sound effects” and “game code” could reside on the game CD and that “other data,” e.g., “data for landscape,” could reside on the webserver. *Id.* at 9:15–19; *see id.* at 9:28–31. Thus, the patent contemplates streaming “code” files for program execution as well as data files.

Limiting “data packs” to “portions of a data file,” as Petitioner proposes, would exclude from claim coverage a specific embodiment disclosed in the ’808 patent. *See* Ex. 1001, 9:8–23; *see also id.* at 2:32–35. A construction excluding a specific embodiment is “rarely, if ever correct.” *PPC Broadband, Inc. v. Corning Optical Commc’ns RF, LLC*, 815 F.3d 747, 755 (Fed. Cir. 2016). “[W]here claims can reasonably [be] interpreted to include a specific embodiment, it is incorrect to construe the claims to exclude that embodiment, absent probative evidence on the contrary.” *Oatey Co. v. IPS Corp.*, 514 F.3d 1271, 1277 (Fed. Cir. 2008). Petitioner identifies no probative evidence that warrants excluding a specific embodiment from claim scope. *See* Pet. 10–11; Reply 7.

Patent Owner proposes a construction for “data packs” that uses the word “asset.” Resp. 14. As discussed above, however, the ’808 patent’s specification equates “data packs” to “assets.” Ex. 1001, 7:1, 7:19–20. So Patent Owner’s proposed construction does not help clarify the meaning of “data packs” and could create a dispute about the meaning of an “asset.”

Hence, based on the intrinsic evidence, we construe “data packs” to mean “portions of a file that has been broken up for streaming.” *See* Inst. Dec. 16–23.

3. “MINIFILTER”

Independent claims 1 and 14 recite “a minifilter associated with a filter manager in an I/O stack.” Ex. 1001, 12:54–55, 14:10–11. Dependent claims 2, 3, 5–7, 11–13, 15–18, and 20 impose various requirements on the “minifilter.” *Id.* at 13:1–8, 13:11–20, 13:38–53, 14:25–39, 14:49–54.

Petitioner proposes that we construe “minifilter” to mean “a filter that is managed such that it is called only for operations for which it has

registered.” Pet. 11 (citing Ex. 1003 ¶ 72). Specifically, Petitioner asserts that the ’808 patent (1) “equates a minifilter with a ‘managed filter’” and (2) “distinguishes legacy filters” that received “each I/O request handled by the stack whether or not the filter would do any work on the request.” *Id.* (citing Ex. 1001, 4:50–52, 5:46–53; Ex. 1003 ¶¶ 73–74; Ex. 1026 ¶ 15). Petitioner also asserts that “the settled meaning of ‘minifilter’ in the art by 2008” comports with its proposed construction. *Id.* at 12–13 (citing Ex. 1003 ¶¶ 76–77; Exs. 1009–1013; Ex. 1026 ¶¶ 13–18).

Patent Owner proposes that we construe “minifilter” to mean “a filter that is managed such that it is associated with a filter manager in an I/O stack called only for operations for which it has registered.” Resp. 14.

For the reasons explained below, we agree with Petitioner that “minifilter” means “a filter that is managed such that it is called only for operations for which it has registered.” *See* Pet. 11.

First, the claim language supports the conclusion that “minifilter” means “a filter that is managed such that it is called only for operations for which it has registered” because claims 1 and 14 recite “a minifilter **associated with** a filter manager in an I/O stack.” Ex. 1001, 12:54–55, 14:10–11 (emphasis added). When a minifilter registers for particular I/O operations with a filter manager, the minifilter becomes “associated with” the filter manager. *See id.* at 2:55–58, 3:26–29, 5:29–32, 6:4–8.

Second, the ’808 patent’s specification supports the conclusion that “minifilter” means “a filter that is managed such that it is called only for operations for which it has registered.” *See* Ex. 1001, 2:55–58, 3:26–29, 4:12–14, 4:50–52, 5:29–32, 6:1–8. In particular, the specification equates a minifilter to a managed filter by explaining that a filter manager

communicates with “a managed filter or minifilter configured to register for I/O requests.” *Id.* at 4:12–14; *see id.* at 2:55–58, 3:26–29, 4:50–52, 5:29–32, 5:46–51, 6:1–4. The specification also explains that a minifilter “attaches to the file system stack indirectly” by “registering with the filter manager” for “the I/O operations the minifilter driver chooses to filter.” *Id.* at 5:29–32; *see id.* at 2:55–58, 3:26–29, 6:4–8. Furthermore, the specification distinguishes a legacy filter from a minifilter by explaining that (1) a legacy filter “pass[es] all I/O requests to the next-lower driver” and (2) a minifilter “registers only for the I/O operations it must handle.” *Id.* at 5:46–51.

Third, the extrinsic evidence indicates the ordinary and customary meaning of “minifilter” at the time of the invention was “a filter that is managed such that it is called only for operations for which it has registered.” *See* Ex. 1003 ¶¶ 50, 76–79; Exs. 1009–1013; Ex. 1026 ¶¶ 13–22, 29–30. For example, a 2004 Microsoft presentation describes a minifilter as “[a] file system filter developed to the Filter Manager model.” Ex. 1011, 8; *see* Ex. 1003 ¶ 77; Ex. 1009, 4 (2004 Microsoft website); Ex. 1010, 3–4 (2004 Microsoft Filter Driver Development Guide); Ex. 1026 ¶ 21. The 2004 Microsoft presentation explains that a minifilter “registers only for operations in which it is interested.” Ex. 1011, 17; *see* Ex. 1013, 12 (2003 Microsoft presentation).

Fourth, Dr. Houh, Mr. Catlin, and Mr. Mason agree that a minifilter registers with a filter manager. *See* Ex. 1003 ¶¶ 72–73, 79 (citing Ex. 1001, 4:50–52, 5:46–53); Ex. 1026 ¶¶ 19, 36 (citing Ex. 1011, 17); Paper 7 ¶ 61.

Patent Owner proposes a construction for “minifilter” that includes the language “associated with a filter manager in an I/O stack.” Resp. 14. But claims 1 and 14 recite “a minifilter associated with a filter manager in an

I/O stack” and already include the language “associated with a filter manager in an I/O stack.” Ex. 1001, 12:54–55, 14:10–11. So Patent Owner’s proposed construction includes duplicative and unnecessary language.

Hence, based on the intrinsic evidence and the extrinsic evidence, we construe “minifilter” to mean “a filter that is managed such that it is called only for operations for which it has registered.” We note that this construction does not render dependent claims 3 and 15 superfluous because each dependent claim additionally requires a minifilter “attaching” to “the I/O stack,” not just “associated with a filter manager” according to claims 1 and 14. *See* Ex. 1001, 12:54–55, 13:4–8, 14:10–11, 14:25–29.

4. “FILTER MINIFILTER DRIVER”

Claim 2 depends directly from claim 1 and recites “said minifilter being configured to load before compression and filter minifilter drivers in the I/O stack.” Ex. 1001, 13:1–3.

Petitioner proposes that we construe “filter minifilter driver” to include “an encryption minifilter driver.” Pet. 16 (citing Ex. 1003 ¶ 89). Specifically, Petitioner asserts that the ’808 patent uses the term “filter minifilter driver” several times but does not explain the term’s meaning. *Id.* (citing Ex. 1001, 2:54–55, 5:39–42, code (57)). Petitioner also asserts that the patent’s Figure 2 depicts “an encryption minifilter as an example of a ‘filter minifilter driver’” by showing encryption minifilter 255. *Id.* (emphasis omitted) (citing Ex. 1003 ¶ 91). Petitioner further asserts that during prosecution the Examiner cited portions of the Robert reference that disclose “placing an antivirus filter before an encryption filter in a stack” as

teaching “compression and filter minifilter drivers in the I/O stack” according to claim 2. *Id.* (citing Ex. 1002, 62; Ex. 1019 ¶¶ 13, 50).

Patent Owner agrees that “filter minifilter driver” includes “an encryption minifilter driver.” *See* Prelim. Resp. 17–28.

Hence, based on agreement between the parties (and for the reasons explained in our Institution Decision), we construe “filter minifilter driver” to include “an encryption minifilter driver.” *See* Inst. Dec. 26–28.

5. “EXTENDED DATA”

Claim 13 depends directly from claim 1 and recites “said minifilter being configured, if the I/O request is a create operation, to attach extended data to a file stream, said extended data corresponding to the create operation.” Ex. 1001, 13:50–53.

Petitioner proposes that we construe “extended data” to mean “additional information associated with the file stream, *i.e.*, other than the data of the file stream itself.” Reply 2; *see* Pet. 18–19. Petitioner asserts that the ’808 patent discloses “name and path information” as an example of “extended data.” Reply 2 (citing Ex. 1001, 7:27–29, Fig. 4 (step 406)).

In the Response, Patent Owner proposes that we construe “extended data” to mean “the name and address in the form recorded in the [memory lookup audit table] MLAT, offset (location) and length (size).” Resp. 12–13 (citing Ex. 1001, 7:23–32, Fig. 4). In the Sur-reply, Patent Owner proposes that we construe “extended data” to mean “data unrelated to the contents of the file stream associated with the file stream by a file system, file system filter driver, or file system mini-filter driver.” Sur-reply 5–6 (citing Ex. 2043, 13–15).

Based on the intrinsic evidence and for the reasons explained below, we construe “extended data” to mean “data related to a file,” such as a file name or address. Starting with the claim language, claim 13 uses the term “extended data” without reciting any details, and no other claim uses that term. Ex. 1001, 12:44–14:54. Hence, the claim language provides little help in determining the meaning of “extended data” to an ordinarily skilled artisan at the time of the invention.

But the ’808 patent’s specification supports the conclusion that “extended data” means “data related to a file,” such as a file name or address. Specifically, the patent discloses a method of processing data requests including read, write, and create operations. *See* Ex. 1001, 3:61–64, 7:23–8:18, Fig. 4. Initially, an I/O stack receives an “application data request” and then determines “the type of request,” e.g., read, write, or create operation. *Id.* at 7:23–25, 7:29–30, 7:61–62, 8:12–13, Fig. 4 (steps 400, 402, and 408). For a read or write operation, a minifilter determines if “the data pack exists.” *Id.* at 7:25–27, Fig. 4 (step 404). For that determination, Figure 4 includes the legend “Does extended data exist?” *Id.* at Fig. 4 (step 404). If “the data pack exists,” the minifilter determines if “the name and path of the data pack sought matches the name and address” stored in a lookup table, e.g., a memory lookup audit table (MLAT). *Id.* at 7:27–29, Fig. 4 (step 406). For that determination, Figure 4 includes the legend “MLAT Name/Path match extended data?” *Id.* at Fig. 4 (step 406). Thus, a read or write operation causes the minifilter to compare the required data pack’s name and path/address (identified as extended data in Figure 4) with a stored name and path/address. *Id.* at 7:25–29, Fig. 4 (steps 404

and 406); *see id.* at 7:29–34, 7:38–40, Fig. 4 (steps 408 and 410); Resp. 12–13; Reply 2–3, 3 n.2.

The '808 patent's prosecution history also supports the conclusion that “extended data” means “data related to a file,” such as a file name or address. In the first Office action, the Examiner rejected application claim 13 (corresponding to patent claim 13) under § 103(a) as obvious over Robert and Poli. Ex. 1002, 60, 64. In doing so, the Examiner cited Robert's paragraphs 94 and 105 as teaching claim 13's “extended data.” *Id.* at 64 (emphasis omitted) (citing Ex. 1019 ¶¶ 94, 105). Those paragraphs disclose a create operation that (1) provides a FILE_OBJECT data structure for file data and (2) associates a file name and a handle (or pointer/address) with the FILE_OBJECT data structure. Ex. 1019 ¶¶ 94, 105; *see* Ex. 1003 ¶ 105; Ex. 1008, 1:67–2:2 (describing a handle); Ex. 1018, 244 (defining a handle); Ex. 2088, 2834 (discussing a FILE_OBJECT data structure). Thus, the Examiner mapped claim 13's “extended data” to a file name and a handle (or pointer/address). *See* Ex. 1002, 64; Ex. 1019 ¶¶ 94, 105. The Applicant did not dispute the Examiner's mapping. *See, e.g.*, Ex. 1002, 76–84; Ex. 1003 ¶ 105.

Hence, based on the intrinsic evidence, we construe “extended data” to mean “data related to a file,” such as a file name or address.

6. “FILE STREAM”

Petitioner proposes that we construe “file stream” in claim 13 to mean “a transmission of data provided from a file.” Reply 5 (citing Ex. 1035 ¶¶ 11–12). Petitioner contends that a “file” is “the basic unit of storage that enables a computer to distinguish one set of information from another,” and Petitioner quotes a Microsoft Computer Dictionary as support for that

definition. *Id.* (quoting Ex. 1033, 211). Petitioner contends that the '808 patent “defines ‘streaming’ to mean simply providing data from a data source.” *Id.* (citing Ex. 1001, 6:37–40). Based on the definition of “file” and the definition of “streaming,” Petitioner asserts that an ordinarily skilled artisan would understand “file stream” to mean “a transmission of data provided from a file.” *Id.*

Patent Owner proposes that we construe “file stream” to mean “open instance of a file or, for a file system that permits multiple data streams within a given file or directory, an open instance of the stream.” Resp. 14 (citing Ex. 2088); *see* Sur-reply 7.

Based on the intrinsic evidence and for the reasons explained below, we construe “file stream” to mean “data structure for file data,” such as a FILE_OBJECT data structure. Starting with the claim language, claim 13 uses the term “file stream” without reciting any details, and no other claim uses that term. Ex. 1001, 12:44–14:54. Hence, the claim language provides little help in determining the meaning of “file stream” to an ordinarily skilled artisan at the time of the invention.

The disclosure in the '808 patent's specification parallels the claim language. *See* Ex. 1001, 3:11–13, 8:15–16, code (57). For instance, the specification states, “If the I/O request is a create operation, the minifilter attaches extended data to a file stream.” *Id.* at 3:11–13, code (57). Hence, the specification provides little help in determining the meaning of “file stream” to an ordinarily skilled artisan at the time of the invention.

But the '808 patent's prosecution history provides more help in determining that meaning. As discussed above, the Examiner rejected application claim 13 (corresponding to patent claim 13) under § 103(a) as

obvious over Robert and Poli. Ex. 1002, 60, 64; *see supra* § III.C.5. In doing so, the Examiner cited Robert’s paragraphs 94 and 105 as teaching claim 13’s “file stream.” Ex. 1002, 64 (emphasis omitted) (citing Ex. 1019 ¶¶ 94, 105). Those paragraphs disclose a create operation that (1) provides a FILE_OBJECT data structure for file data and (2) associates a file name and a handle (or pointer/address) with the FILE_OBJECT data structure. Ex. 1019 ¶¶ 94, 105; *see* Ex. 1003 ¶ 105; Ex. 1008, 1:67–2:2 (describing a handle); Ex. 1018, 244 (defining a handle); Ex. 2088, 2834 (discussing a FILE_OBJECT data structure). Thus, the Examiner mapped claim 13’s “file stream” to a FILE_OBJECT data structure for file data. *See* Ex. 1002, 64; Ex. 1019 ¶¶ 94, 105. The Applicant did not dispute the Examiner’s mapping. *See, e.g.*, Ex. 1002, 76–84; Ex. 1003 ¶ 105.

Hence, based on the intrinsic evidence, we construe “file stream” to mean “data structure for file data,” such as a FILE_OBJECT data structure.

7. OTHER CLAIM LANGUAGE

We determine that no other claim term or phrase requires an explicit construction to decide the patentability issues for claims 1–20. “[O]nly those terms need be construed that are in controversy, and only to the extent necessary to resolve the controversy.” *Vivid Techs., Inc. v. Am. Sci. & Eng’g, Inc.*, 200 F.3d 795, 803 (Fed. Cir. 1999); *see Nidec Motor Corp. v. Zhongshan Broad Ocean Motor Co.*, 868 F.3d 1013, 1017 (Fed. Cir. 2017).

D. Alleged Obviousness over Christiansen and Eylon: Claims 1–20

Petitioner contends that claims 1–20 are unpatentable under § 103(a) as obvious over Christiansen and Eylon. *See* Pet. 5, 30–65. Below, we provide overviews of Christiansen and Eylon, and then we consider the

obviousness issues raised by the parties. For the reasons explained below, we agree with Petitioner that claims 1–12 and 14–20 are unpatentable under § 103(a) as obvious over Christiansen and Eylon but disagree that claim 13 is unpatentable based on these references.

1. OVERVIEW OF CHRISTIANSEN (EXHIBIT 1006)

Christiansen is a U.S. patent application publication titled “Method and System for Caching Remote Files Locally,” filed on November 30, 2004, and published on June 1, 2006. Ex. 1006, codes (22), (43), (54). Christiansen explains that providing content servers “with all the content of a datacenter, and keeping the content on them up-to-date so that they can serve any content requested is expensive both in storage and transmission costs.” *Id.* ¶ 2. Christiansen identifies a need for “effectively caching remote data locally such that the entirety of the content set can be stored in separate dedicated storage while a subset of the content is cached on the content server hard disk.” *Id.* ¶ 3.

Christiansen's Figure 2 is reproduced below:

FIG. 2

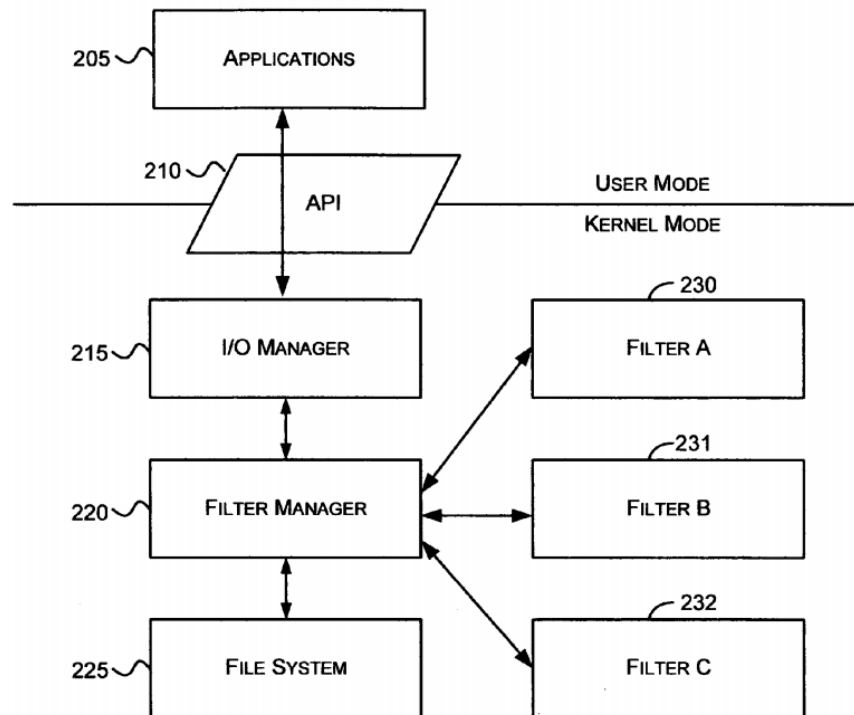


Figure 2, above, “is a block diagram representing an exemplary arrangement of components of a system” for practicing Christiansen’s invention.

Ex. 1006 ¶¶ 10, 32, Fig. 2.

As Figure 2 shows, the system includes the following components: applications 205, application programming interface (API) 210, input/output (I/O) manager 215, filter manger 220, file system 225, filter 230 (filter A), filter 231 (filter B), and filter 232 (filter C). Ex. 1006 ¶ 32, Fig. 2. A filter may take “any set of actions” performed by “a kernel-mode or user-mode process.” *Id.* ¶ 39; *see id.* ¶ 43. Moreover, a filter may operate “reactive[ly] (e.g., wait until it receives I/O requests before acting) and/or proactive[ly]

(e.g., initiate its own I/O requests or perform other actions asynchronously with I/O requests handled by the I/O manager 215).” *Id.* ¶ 39.

An application makes a file system request through API 210 to I/O manager 215. Ex. 1006 ¶ 33. After receiving the file system request, I/O manager 215 “determine[s] what I/O request or requests should be issued to fulfill” the file system request and “send[s] each I/O request” to filter manager 220. *Id.*

Filters 230, 231, and 232 register with filter manager 220, e.g., during an initialization routine. Ex. 1006 ¶¶ 32, 34. Typically, a filter registers only for particular I/O operations, i.e., I/O operations “it may be interested in processing,” such as one or more “create, read, write, close, [and] rename” operations. *Id.* ¶ 34. A create operation “may create an object or open an already-existing object.” *Id.* ¶ 60.

As an example, “an encryption filter may register for read and write I/Os, but not for others wherein data does not need to be encrypted or decrypted.” Ex. 1006 ¶ 34. As another example, “a quota filter may be interested only in object creates and object writes.” *Id.*

A filter may register for pre-callbacks or post-callbacks (or both). Ex. 1006 ¶ 35. A “pre-callback is called as data associated with an I/O request propagates from the I/O manager 215 towards the file system 225.” *Id.* A “post-callback is called during the completion of the I/O request as data associated with the I/O request propagates from the file system 225 towards the I/O manager 215.” *Id.* Furthermore, a filter may “specify whether the filter should be notified” for a pre-callback or a post-callback (or both) for each type of I/O operation that the filter has registered with filter manager 220. *Id.*

Filter manager 220 may “create a data structure” for the callback data. Ex. 1006 ¶ 36. Filter manager 220 may “call and pass the callback data to each filter that has registered to receive callbacks for the type of I/O received by the filter manager.” *Id.*; *see id.* ¶ 37. Filter manager 220 passes the callback data “associated with a particular type of I/O request to each registered filter sequentially in an order in which the registered filters are ordered.” *Id.* ¶ 37. For example, if filter 230 precedes filter 232 in processing I/O requests and if filter manager 220 receives a read I/O request, “the filter manager 220 may first call and pass the callback data to the filter 230 and after the filter 230 has processed the callback data, the filter manager 220 may then call and pass the callback data (as modified, if at all) to the filter 232.” *Id.* But a filter “may stop an I/O request from propagating further” and “may store data in memory and persist (e.g., store) this data on disk.” *Id.* ¶ 39.

Christiansen’s Figure 5 is reproduced below:

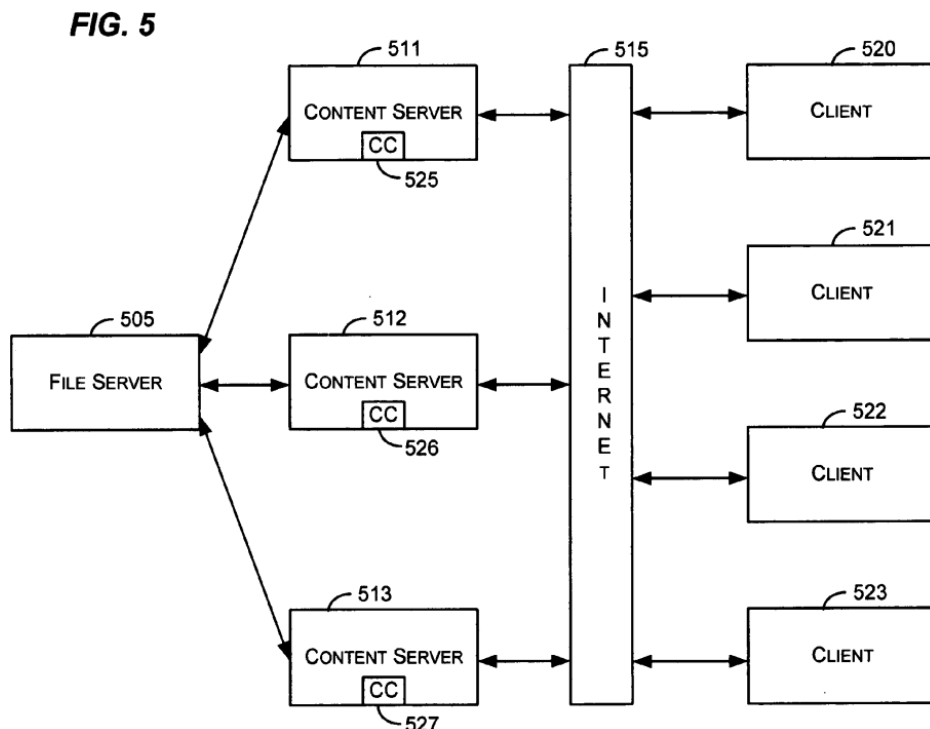


Figure 5, above, “is a block diagram representing an exemplary environment” for practicing Christiansen’s invention. Ex. 1006 ¶¶ 13, 45, Fig. 5.

As Figure 5 shows, the environment includes file server 505, content servers 511, 512, and 513, and clients 520, 521, 522, and 523. Ex. 1006 ¶ 45, Fig. 5. “The various entities may communicate with each other via various networks including intra-networks and the Internet 515.” *Id.* ¶ 45. File server 505 “may include a set of all objects (e.g., directories, files, other content, and the like) that may be available to clients from a datacenter.” *Id.* ¶ 46. Content servers 511, 512, and 513 “may access these objects when providing content to a client.” *Id.* After accessing an object from file server 505, “a content server may then provide the object to a client.” *Id.*

Content servers 511, 512, and 513 may contain caching components 525, 526, and 527, respectively, and “may or may not cache objects obtained” from file server 505. Ex. 1006 ¶ 46. Each caching component “may include various subcomponents including a filter and caching service.” *Id.* ¶ 48.

Christiansen's Figure 7 is reproduced below:

FIG. 7

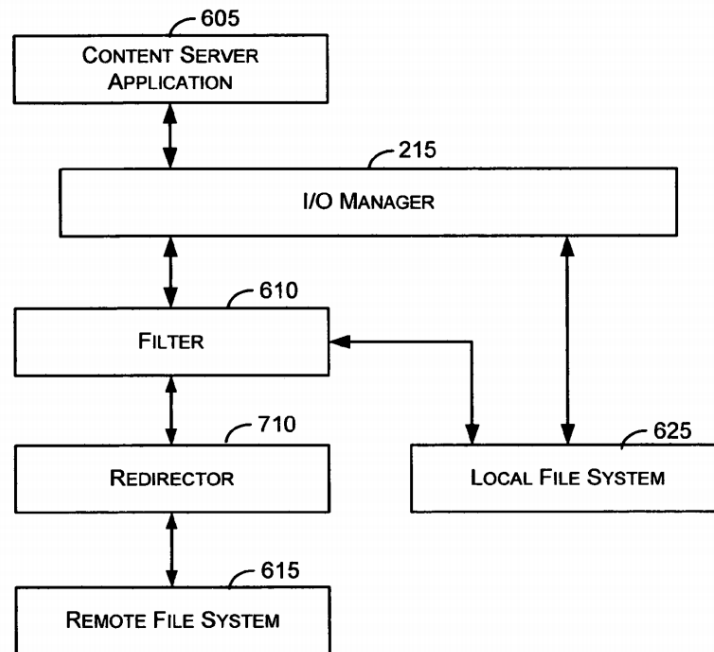


Figure 7, above, “is a block diagram representing an exemplary arrangement of components of a system” for practicing Christiansen’s invention.

Ex. 1006 ¶¶ 15, 55, Fig. 7.

As Figure 7 shows, the system includes the following components: I/O manager 215, content server application 605, filter 610, remote file system 615, local file system 625, and redirector 710. Ex. 1006 ¶ 55, Fig. 7. Content server application 605 may “independently access objects from the remote and local file systems 615 and 625.” *Id.* ¶ 50. Filter 610 may include caching functionality. *Id.* ¶ 52.

For instance, filter 610 may obtain “a mapping of remote file system names (e.g., network shares) to local cache directory names.” Ex. 1006 ¶ 57. Filter 610 may store the mapping in a mapping table maintained in

memory. *Id.* ¶¶ 57, 68; *see id.* ¶¶ 58, 67, Fig. 9. Filter 610 may use the mapping table to “determine when to redirect requests for objects located remotely to local cache directories.” *Id.* ¶ 57; *see id.* ¶ 54.

Furthermore, filter 610 may “monitor I/Os to and from the remote file system 615.” Ex. 1006 ¶ 50; *see id.* ¶ 6. When filter 610 “receives a request to open a remote object,” filter 610 may use the mapping table to “determine whether the object is cached on the local file system 625.” *Id.* ¶ 58; *see id.* ¶¶ 4, 68, code (57), Fig. 9. If “the object is cached on the local file system 625,” filter 610 may “instruct the I/O manager 215 to redirect the request to the local file system 625.” *Id.* ¶ 58; *see id.* ¶¶ 4, 69–70, code (57), Fig. 9.

2. OVERVIEW OF EYLON (EXHIBIT 1005)

Eylon is a U.S. patent titled “Method and System for Executing Network Streamed Application,” filed on December 28, 2000, and issued on June 3, 2003. Ex. 1005, codes (22), (45), (54). Eylon states that the invention concerns “streaming applications from a server for execution on a client.” *Id.* at 1:35–37.

Eylon's Figure 1 is reproduced below:

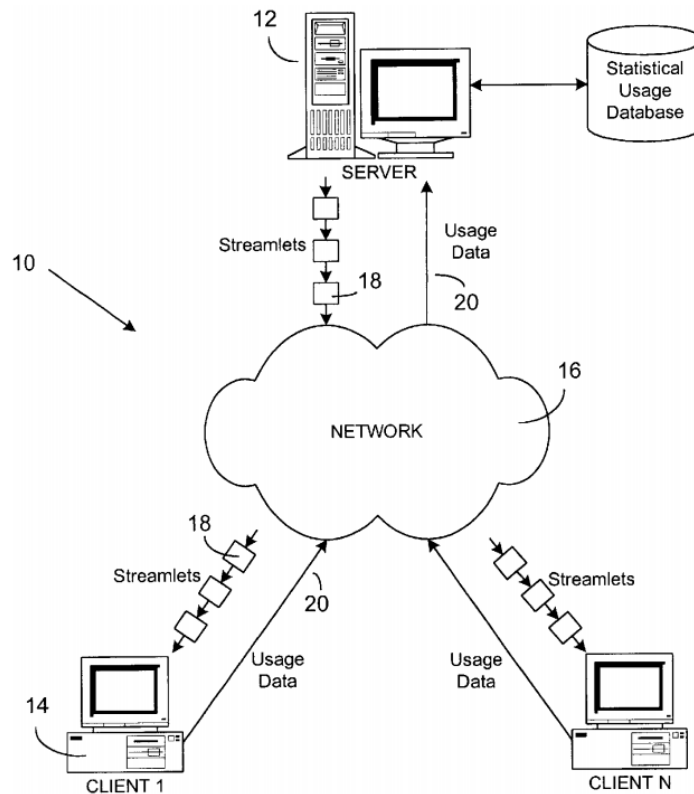


FIG. 1

Figure 1, above, “is a block diagram of a system” for streaming applications from a server for execution on a client. Ex. 1005, 5:26–27, 5:46–47, Fig. 1.

As Figure 1 shows, system 10 includes server 12 connected to one or more clients 14 via “data network 16, such as the Internet, an intranet, extranet or other TCP/IP based communication network, or other types of data networks, including wireless data networks.” Ex. 1005, 5:48–52.

Before streaming an application to a client, the server stores the application’s files as multiple streamlets, i.e., “parts into which the application has been divided.” *Id.* at 3:57–59, 5:53–54, code (57). In a preferred embodiment, the streamlets correspond to data blocks that a client

would process with its “native operating system” if the entire application was present locally at the client. *Id.* at 3:59–63; *see id.* at 12:1–3.

When a client seeks to execute an application, the server delivers an “initial set of streamlets” to the client. Ex. 1005, 4:42–45, code (57); *see id.* at 3:52–56, 5:65–6:4, 6:18–21. The “initial set of streamlets” may comprise “a small portion of the application.” *Id.* at 5:54–57; *see id.* at 3:52–56, 6:21–25. After the client receives the “initial set of streamlets,” the client begins executing the application. *Id.* at 3:52–56, 5:65–6:4, 6:18–21, 9:59–61.

The server also delivers a file-structure specification to the client. Ex. 1005, 5:54–64. The file-structure specification “defines how files associated with the application and required for the application to operate appear to a computer” if the entire application was present locally at the client. *Id.* at 5:57–61; *see id.* at 9:65–10:1.

Eylon’s Figure 4 is reproduced below:

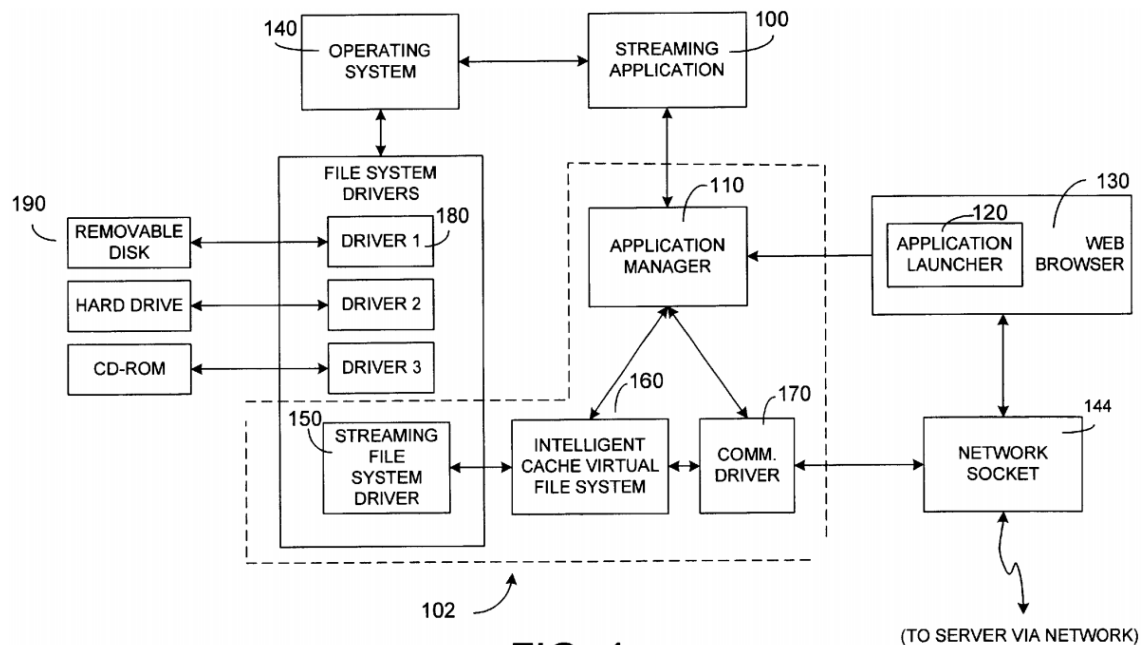


FIG. 4

Figure 4, above, “is a block diagram of the client system showing various elements of [a] streaming control system.” Ex. 1005, 5:32–33, 8:1–4, Fig. 4.

As Figure 4 shows, client operating system 140 has “access to one or more data storage devices 190, such as a removable disk, a hard drive, and a CD-ROM.” Ex. 1005, 8:7–9, Fig. 4. “Each storage device 190 typically has an associated device driver 180 which acts as an interface between the standard I/O protocols used by the operating system 140 and the specific I/O protocols used by the given device 190.” *Id.* at 8:10–13.

As Figure 4 also shows, streaming support system 102 includes application manager 110, file system driver (FSD) 150, virtual file system (VFS) 160, and communication driver 170. Ex. 1005, 8:14–41, Fig. 4; *see id.* at 4:1–36. FSD 150 acts as an interface between client operating system 140 and VFS 160. *Id.* at 8:20–23. VFS 160 “store[s] and organize[s] received program streamlets” and appears to client operating system 140 as “a local virtual file system which can be accessed by the operating system in the same manner as other data storage devices.” *Id.* at 8:14–18, 8:23–27; *see id.* at 11:31–34.

Streaming support system 102 may perform several functions including the following:

- (1) check “for the availability of a valid client environment”;
- (2) “initiate the streaming of streamlets from the server”;
- (3) “process received streamlets”;
- (4) “initiate execution of a streaming application after sufficient streamlets have been received”;
- (5) “request specific streamlets from the server when necessary”;

- (6) register a streaming application “with the VFS 160 and/or FSD 150 to limit access to the streamed data to authorized applications as a way of preventing a user from extracting streamed data from the VFS 160 without authorization”;
- (7) “monitor and meter the usage of the program”;
- (8) “indicate the application status to the end user”; and
- (9) “perform or initiate cleanup functions when the application” terminates.

Ex. 1005, 7:58–67, 8:41–53, 8:62–64, 9:28–33; *see id.* at 17:47–59.

Eylon’s Figure 7 is reproduced below:

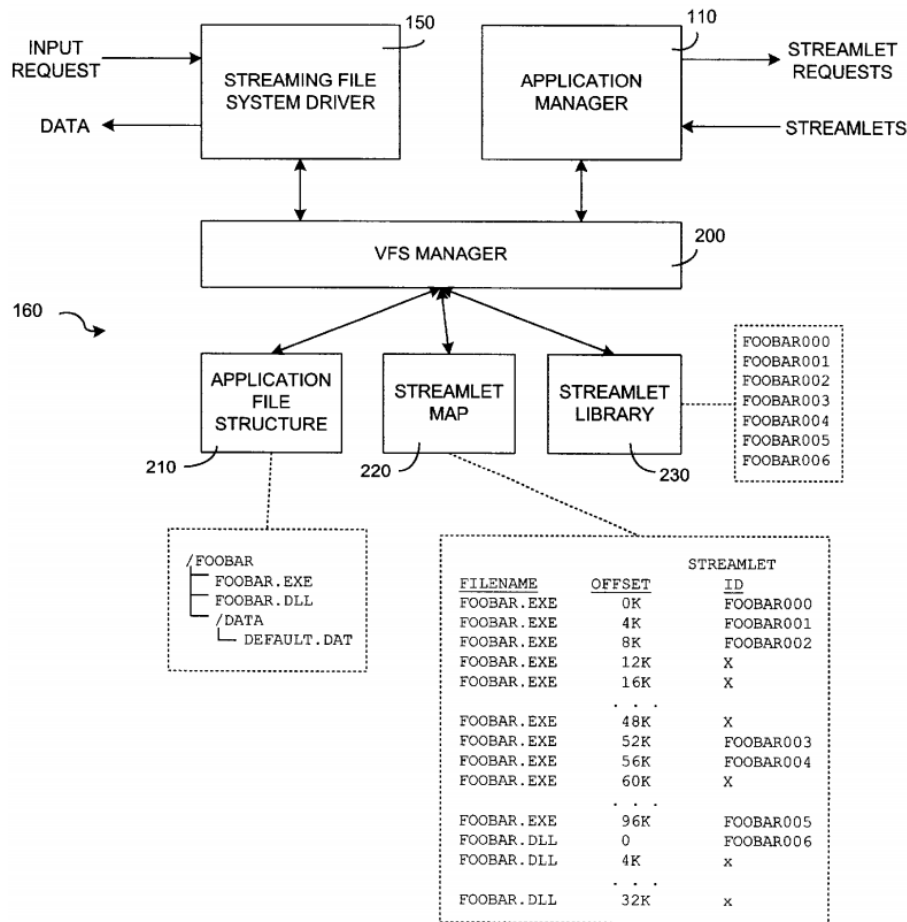


FIG. 7

Figure 7, above, “is a block diagram of one embodiment of a virtual file system containing a partially streamed application.” Ex. 1005, 5:38–39, 11:31–34, Fig. 7.

As Figure 7 shows, VFS 160 includes VFS manager 200 and the following data or library files that “contain the streamed data and information needed to map that data” to each streaming application: file structure library 210, streamlet map 220, and streamlet library 230. Ex. 1005, 11:34–44, 11:64–67, Fig. 7. File structure library 210 contains “the complete folder/file structure for each streaming application.” *Id.* at 11:38–40, 11:65–67, Fig. 7. Streamlet library 230 contains the streamlets received from the server. *Id.* at 11:40–42, 12:32–37, Fig. 7. Streamlet map 220 associates “the various streamlets in the library 230 with a respective file name and position within that file.” *Id.* at 11:42–44, 12:9–13, Fig. 7. Thus, VFS 160 “knows the framework, files, folder/directory structure, etc.” of each streaming application. *Id.* at 12:52–57.

While Figure 7 shows VFS 160 including VFS manager 200 with file structure library 210, streamlet map 220, and streamlet library 230, Eylon explains that the “actual mapping and local streamlet storage can be done in a variety [of] ways using various database and linked object mechanisms and data structures.” Ex. 1005, 12:22–24. Eylon also explains that the various mechanisms and data structures “are known to those of skill in the art.” *Id.* at 12:23–25.

3. INDEPENDENT CLAIM 1

(a) Preamble [1.a]

Claim 1’s preamble recites “[a] streaming on demand system.” Ex. 1001, 12:44.

Petitioner contends that Eylon teaches claim 1's preamble because Eylon's streaming system handles "on-demand request[s]" from users. Pet. 30–31 (alteration by Petitioner) (emphasis omitted) (citing Ex. 1005, 2:45–48, 3:1–4, 7:4–8, code (57)). Specifically, Petitioner asserts that Eylon discloses that when "the user strays from the predictive path and an on-demand request is issued, the predictive routine can be re-centered on the appropriate node associated with the on-demand request and the predictive stream restarted from that point." *Id.* at 31 (emphases omitted) (quoting Ex. 1005, 7:4–8).

Patent Owner disputes the Eylon teaches claim 1's preamble. *See* Resp. 24–25; Sur-reply 13–14. Specifically, Patent Owner asserts that in Eylon's system a server pushes data to a client based on a "predictive routine." Resp. 24–25 (citing Ex. 1005, 7:4–8). Patent Owner asserts that "Eylon prohibits on-demand streaming." *Id.* at 25 (citing Ex. 1005, 14:3–54). Patent Owner also asserts that no "transactions initiated in response to requests (i.e., demands) by the client" are based on "client-initiated predictive streaming." Sur-reply 14 (emphasis omitted).

Generally, a preamble does not limit a claim. *Allen Eng'g Corp. v. Bartell Indus., Inc.*, 299 F.3d 1336, 1346 (Fed. Cir. 2002). Here, we need not decide whether claim 1's preamble limits the claim because we agree with Petitioner that Eylon teaches the preamble. *See* Pet. 30–31; Ex. 1003 ¶¶ 148–151; Ex. 1035 ¶ 17.

Eylon explains that "[p]referably, the server is configured to automatically forward sequences of streamlets to the client using a predictive streaming engine" that "selects streamlets to forward according to [a] dynamic statistical knowledge base generated by analyzing the various

sequences in which the application program attempts to load itself into memory as various program features are accessed.” Ex. 1005, 6:43–49; *see id.* at 4:37–48, code (57); Ex. 1003 ¶ 173; Ex. 1035 ¶ 17. But Eylon also explains that when “the user strays from the predictive path,” the client issues an on-demand request (e.g., a fetch request) to the server. Ex. 1005, 6:25–27, 6:37–39, 7:4–8; *see id.* at 13:35–37; Ex. 1003 ¶ 149. Additionally, Eylon discloses that the client may “initiate[] a request for additional streamlets” in response to “user input.” Ex. 1005, 12:43–51; *see* Ex. 1003 ¶ 150; Ex. 1035 ¶ 17. Depending on “user input,” the client may “request specific streamlets from the server when necessary.” Ex. 1005, 7:56–64; *see* Ex. 1003 ¶¶ 156–158; Ex. 1035 ¶ 17. Eylon also discloses that the server services the client’s on-demand requests. Ex. 1005, 14:51–54; *see id.* at 7:56–67, 11:1–10.

Thus, contrary to Patent Owner’s assertion, a server not only pushes data to a client using a predictive streaming engine, the server also services the client’s on-demand requests. Ex. 1005, 7:4–8, 7:56–64, 12:43–51, 14:51–54; *see* Ex. 1035 ¶ 17; Resp. 24–25. Hence, as Petitioner contends, Eylon’s streaming system handles “on-demand request[s]” from users. Ex. 1005, 7:4–8; *see id.* at 2:45–48, 3:1–4, 3:45–5:18, 6:37–39, 7:56–64, 11:1–10, 12:43–51, 13:35–37, 14:51–54, codes (54), (57); Ex. 1003 ¶¶ 148–151; Ex. 1035 ¶¶ 15–18; Pet. 30–31.

As for Patent Owner’s assertion that no “transactions initiated in response to requests (i.e., demands) by the client” are based on “client-initiated predictive streaming,” nothing in claim 1 requires “client-initiated predictive streaming.” *See* Ex. 1001, 12:44–67; Sur-reply 14. Additionally, Patent Owner identifies no disclosure in the ’808 patent that would support a

requirement for “client-initiated predictive streaming” according to § 112’s first paragraph. *See, e.g.*, Resp. 24–25; Sur-reply 13–14.

(b) Limitation [1.b.i]

Claim 1 recites “an on-demand requester object installed on a computing device.” Ex. 1001, 12:44–45.

Petitioner contends that Eylon teaches this limitation because Eylon discloses streaming support system 102 installed on a client device, e.g., a personal computer. *See* Pet. 31 (citing Ex. 1005, 7:56–61, 9:4–7). Specifically, Petitioner asserts that streaming support system 102 corresponds to the claimed “on-demand requester object” because “it is [a] discrete portion of machine-readable code configured to initiate on-demand streaming, by requesting streamlets from a remote source as needed, in response to application I/O requests.” *Id.* at 32 (citing Ex. 1003 ¶ 159; Ex. 1005, 4:1–22, 7:4–8, 7:56–67).

Patent Owner disputes that Eylon teaches limitation [1.b.i]. *See* Resp. 25–26; Sur-reply 14–16. Specifically, Patent Owner asserts that “a significant part of the Eylon system must be stored on and operates from the server to provide the push transmissions as dictated by a predictive engine.” Resp. 25 (citing Ex. 1005, 4:37–56). Patent Owner asserts that Eylon prohibits pull transactions. *Id.* at 26 (citing Ex. 1005, 11:31–55; Ex. 2043, 26–27). Patent Owner also asserts that Petitioner points to “no instances” of “client-initiated predictive queuing” in Eylon. Sur-reply 14.

Additionally, Patent Owner contends that Christiansen concerns “dispersing content from a single unified server farm” and that “[m]oving the content server into the client device would not have been possible for the embodiments described in Christiansen.” Sur-reply 16.

We agree with Petitioner that Eylon teaches limitation [1.b.i]. *See* Pet. 31–32; Ex. 1003 ¶¶ 155–159. Specifically, Eylon discloses streaming support system 102 installed on a client device, e.g., a personal computer. Ex. 1005, 7:56–8:4, 8:39–41, 9:4–7, Figs. 3–4; *see id.* at 4:1–36, 8:18–19; Ex. 1003 ¶¶ 152–159. Streaming support system 102 includes application manager 110, file system driver (FSD) 150, virtual file system (VFS) 160, and communication driver 170. Ex. 1005, 8:14–41, Fig. 4; *see id.* at 4:1–36; Ex. 1003 ¶¶ 120, 158.

FSD 150 acts as an interface between client operating system 140 and VFS 160. Ex. 1005, 8:20–23. VFS 160 “store[s] and organize[s] received program streamlets” and appears to client operating system 140 as “a local virtual file system which can be accessed by the operating system in the same manner as other data storage devices.” *Id.* at 8:14–18, 8:23–27; *see id.* at 11:31–34. Specifically, VFS 160 “implements a sparsely populated file system” where VFS 160 “knows the framework, files, folder/directory structure, etc., which would exist if the application were fully installed, but (unknown to the operating system) the actual file contents are only partially existent in local storage.” *Id.* at 12:52–57; *see id.* at 4:1–9, 6:33–37.

Thus, in the proposed combination Eylon’s streaming support system 102 would have been “installed on a computing device” according to limitation [1.b.i]. *See* Ex. 1003 ¶¶ 155–159.

When disputing that Eylon teaches limitation [1.b.i], Patent Owner does not address Eylon’s streaming support system 102 installed on a client device, e.g., a personal computer. *See* Resp. 25–26; Sur-reply 14–16; Ex. 1003 ¶¶ 155–159; Ex. 1005, 4:1–36, 7:56–8:4, 8:18–19, 8:39–41, 9:4–7, Figs. 3–4.

As for Patent Owner’s assertion that Petitioner points to “no instances” of “client-initiated predictive queuing” in Eylon, nothing in claim 1 requires “client-initiated predictive queuing.” *See* Ex. 1001, 12:44–67; Sur-reply 14. Additionally, Patent Owner identifies no disclosure in the ’808 patent that would support a requirement for “client-initiated predictive queuing” according to § 112’s first paragraph. *See, e.g.*, Resp. 25–26; Sur-reply 14–16.

As for Patent Owner’s contention that “[m]oving the content server into the client device would not have been possible for the embodiments described in Christiansen,” that contention does not address the proposed combination because the proposed combination does not involve moving a server into a client device. *See* Sur-reply 16. Instead, Petitioner asserts that in the proposed combination Eylon’s streaming functionality, e.g., streaming support system 102, would have been implemented as a managed filter in Christiansen’s system, e.g., as one of filters 230, 231, and 232 in Christiansen’s Figure 2. Pet. 24–28, 42, 45; *see* Reply 20 & n.12.

(c) Limitation [1.b.ii]

Claim 1 recites “said on-demand requester object being configured to receive I/O requests on behalf of an application for which data is available in data packs for streaming delivery.” Ex. 1001, 12:45–48.

Petitioner contends that Eylon teaches this limitation for several related reasons. *See* Pet. 32–33. Petitioner asserts that Eylon discloses that before streaming “the application files are broken up into streamlets corresponding generally to various portions of the application files and these streamlets are delivered to the client by the server.” *Id.* at 33 (emphasis omitted) (quoting Ex. 1005, code (57)). According to Petitioner, Eylon’s

streamlets correspond to the claimed “data packs.” *Id.* (citing Ex. 1005, 3:57–63, 11:56–67, 12:1–5, Fig. 7). Petitioner also asserts that Eylon “discloses that ‘as the streaming application executes and paging or data I/O requests are generated to retrieve require [sic] code or data, the operating system will automatically direct it to the FSD driver which then passes it in the proper format to the VFS caching system for processing.’” *Id.* at 32 (alteration by Petitioner) (quoting Ex. 1005, 4:17–22).

Patent Owner disputes that Eylon teaches limitation [1.b.ii]. *See* Resp. 26. Specifically, Patent Owner asserts that Eylon “does not receive I/O requests on behalf of the application to initiate streaming delivery.” *Id.*; *see id.* at 29. Patent Owner also asserts that in Eylon’s system a client operating system initiates streaming by generating “a page fault exception” after searching VFS 160. *Id.* at 26 (citing Ex. 1005, 10:13–67; Ex. 2043, 42–45).

We agree with Petitioner that Eylon teaches limitation [1.b.ii]. *See* Pet. 32–33; Ex. 1003 ¶¶ 160–165; Ex. 1035 ¶ 19. Specifically, Eylon discloses breaking an application’s files into multiple streamlets, i.e., “parts into which the application has been divided.” Ex. 1005, 3:57–59, code (57); *see* Ex. 1003 ¶ 162. In a preferred embodiment, the streamlets correspond to data blocks that a client would process with its “native operating system” if the entire application was present locally at the client. Ex. 1005, 3:59–63; *see id.* at 12:1–3; Ex. 1003 ¶ 162. Hence, Eylon’s streamlets are “portions of a file that has been broken up for streaming” as we have construed “data packs.” *See supra* § III.C.2.

In addition, as discussed above, Eylon’s streaming support system 102 includes FSD 150 and VFS 160. *See* Ex. 1005, 8:14–41, Fig. 4; Ex. 1003

¶¶ 120, 158; *supra* § III.D.3(b). FSD 150 acts as an interface between client operating system 140 and VFS 160. Ex. 1005, 8:20–23. VFS 160 “store[s] and organize[s] received program streamlets.” *Id.* at 8:14–16.

Eylon discloses that “as the streaming application executes and paging or data I/O requests are generated to retrieve require[d] code or data, the operating system will automatically direct” the paging and data I/O requests to FSD 150 in streaming support system 102. Ex. 1005, 4:17–20, 8:20–29, Fig. 4; *see id.* at 10:34–42, Fig. 6A; Ex. 1003 ¶¶ 123, 160. For instance, to “open audio files for playblack [sic], and/or open video files for playback,” the streaming application generates data I/O requests that client operating system 140 directs to FSD 150. *See* Ex. 1005, 4:17–20, 8:20–27, Fig. 4; Ex. 1035 ¶ 19. After receiving a paging or data I/O request, FSD 150 passes the I/O request “in the proper format” to VFS 160 in streaming support system 102 for processing. Ex. 1005, 4:20–22, 8:20–29, Fig. 4; *see id.* at 10:43–46, Fig. 6A; Ex. 1003 ¶¶ 123, 160.

Thus, in the proposed combination Eylon’s streaming support system 102 would have been “configured to receive I/O requests on behalf of an application for which data is available in data packs for streaming delivery” according to limitation [1.b.ii]. *See* Ex. 1003 ¶¶ 160–165; Ex. 1035 ¶ 19.

Patent Owner acknowledges that Eylon discloses generating “paging or data I/O requests” to “retrieve code or data.” Resp. 15. When disputing that Eylon teaches limitation [1.b.ii], however, Patent Owner does not address data I/O requests directed to streaming support system 102. *See id.* at 26. Instead, Patent Owner addresses only paging I/O requests. *Id.*

(d) Limitation [1.b.iii]

Claim 1 recites “said on-demand requester object being responsive to application I/O requests.” Ex. 1001, 12:48–49.

Petitioner contends that Eylon teaches this limitation because Eylon discloses that data I/O requests “are automatically directed” to FSD 150 in streaming support system 102, and FSD 150 passes data I/O requests “in the proper format” to VFS 160 in streaming support system 102 for processing. *See* Pet. 34 (citing Ex. 1005, 4:17–22); Reply 11–12 (citing Ex. 1005, 4:15–22). According to Petitioner, Eylon’s processing may include “returning data to the operating system, issuing a fetch request to the server for the appropriate streamlet, or storing streamlet data.” Pet. 34 (citing Ex. 1003 ¶¶ 169–170; Ex. 1005, 4:23–36, 12:43–51, code (57)).

Patent Owner disputes that Eylon teaches limitation [1.b.iii]. *See* Resp. 26–28. Specifically, Patent Owner asserts that in Eylon’s system a client operating system issues paging I/O requests—not application I/O requests—responsive to “page faults when data or code cannot be drawn from a virtual file system.” *Id.* at 26–28 (citing Ex. 1005, 4:10–63).

We agree with Petitioner that Eylon teaches limitation [1.b.iii]. *See* Pet. 34; Reply 11–12; Ex. 1003 ¶¶ 167–171; Ex. 1035 ¶ 20. Specifically, as discussed above, Eylon discloses that “as the streaming application executes and paging or data I/O requests are generated to retrieve require[d] code or data, the operating system will automatically direct” the paging and data I/O requests to FSD 150 in streaming support system 102. Ex. 1005, 4:17–20, 8:20–29, Fig. 4; *see id.* at 10:34–42, Fig. 6A; Ex. 1003 ¶¶ 123, 160, 168; *supra* § III.D.3(c). After receiving a paging or data I/O request, FSD 150 passes the I/O request “in the proper format” to VFS 160 in streaming

support system 102 for processing. Ex. 1005, 4:20–22, 8:20–29, Fig. 4; *see id.* at 10:43–46, Fig. 6A; Ex. 1003 ¶¶ 123, 160, 168; *supra* § III.D.3(c).

If “the requested streamlets are present” in VFS 160, streaming support system 102 returns the data to client operating system 140, and “the streaming program continues normal operation.” Ex. 1005, 4:23–25; *see id.* at 10:46–47, 16:66–67, 17:47–55, Fig. 6A; Ex. 1003 ¶¶ 124, 168.

In Eylon’s system, a streaming application generates data I/O requests that correspond to the claimed “application I/O requests.” Ex. 1005, 4:17–20; *see* Ex. 1035 ¶ 20. For instance, Eylon discloses a streaming application with a data file called DEFAULT.DAT that the streaming application accesses through data I/O requests. Ex. 1005, 5:38–39, 11:57–62, Fig. 7; *see id.* at 4:17–20, 8:20–29; Ex. 1003 ¶ 126; Ex. 1035 ¶¶ 19–20. Streaming support system 102 responds to a data I/O request from a streaming application by providing data to client operating system 140 if “the requested streamlets are present” in VFS 160. Ex. 1005, 4:23–25; *see id.* at 17:47–55; Ex. 1003 ¶¶ 124, 168; Ex. 1035 ¶ 20.

Thus, in the proposed combination Eylon’s streaming support system 102 would have been “responsive to application I/O requests” according to limitation [1.b.iii]. *See* Ex. 1003 ¶¶ 167–171.

As discussed above, Patent Owner acknowledges that Eylon discloses generating “paging or data I/O requests” to “retrieve code or data.” Resp. 15; *see supra* § III.D.3(c). When disputing that Eylon teaches limitation [1.b.iii], however, Patent Owner does not address data I/O requests directed to streaming support system 102. *See* Resp. 26–28.

(e) Limitation [1.b.iv]

Claim 1 recites “said application I/O requests corresponding to sequential and non-sequential data packs.” Ex. 1001, 12:50–51.

Petitioner contends that Eylon teaches this limitation because Eylon discloses that the server may either (1) “automatically forward sequences of streamlets to the client using a predictive streaming engine” or (2) “stray[] from the predictive path” in response to an on-demand request. *See* Pet. 34–35 (alteration by Petitioner) (quoting Ex. 1005, 6:43–45, 7:4–5). Petitioner also contends that the predicted sequences of streamlets correspond to the claimed “sequential” data packs and that the on-demand streamlets correspond to the claimed “non-sequential” data packs. *Id.* at 35.

Patent Owner “does not dispute that Eylon can receive pushed content from the server in either sequential or nonsequential fashion.” Resp. 28.

We agree with Petitioner that Eylon teaches limitation [1.b.iv]. *See* Pet. 34–35; Ex. 1003 ¶¶ 172–175. Specifically, Eylon discloses that the server may “automatically forward sequences of streamlets to the client using a predictive streaming engine” that “selects streamlets to forward according to [a] dynamic statistical knowledge base generated by analyzing the various sequences in which the application program attempts to load itself into memory as various program features are accessed.” Ex. 1005, 6:43–49; *see id.* at 4:37–48, 7:9–25, 7:48–52, 10:47–49; Ex. 1003 ¶ 173. Eylon also discloses that when “the user strays from the predictive path,” the client issues an on-demand request (e.g., a fetch request) to the server, and the server responds to the on-demand request with appropriate streamlets. Ex. 1005, 6:25–27, 6:37–39, 7:4–8; *see id.* at 4:25–28, 9:57–59, 10:53–56, 11:1–10, 13:2–4, 13:35–37; Ex. 1003 ¶ 149. Eylon explains that the server

may provide the client with streamlets “individually or grouped together and pushed to the client in clusters.” Ex. 1005, 6:30–32. Hence, the predicted sequences of streamlets correspond to the claimed “sequential” data packs, and the on-demand streamlets correspond to the claimed “non-sequential” data packs. *See* Ex. 1003 ¶¶ 172–175.

Additionally, Eylon discloses an example where the server provides the client with a startup block containing “28 k of data in seven 4 k streamlets.” Ex. 1005, 9:46–49, 12:5–46; *see id.* at 6:4–17. The client stores the startup block in VFS 160. *Id.* at 12:31–32, Fig. 7; *see id.* at 6:18–30, 11:62–67. In the startup block, six streamlets comprise part of file FOOBAR.EXE, and one streamlet comprises part of file FOOBAR.DLL. *Id.* at 12:5–16, 12:43–46, Figs. 7–8. The six streamlets comprising part of file FOOBAR.EXE have offset addresses starting at 0K, 4K, 8K, 52K, 56K, and 96K, respectively. *Id.* at 12:10–16, Figs. 7–8. For file FOOBAR.EXE, the three streamlets having offset addresses starting at 0K, 4K, and 8K correspond to the claimed “sequential” data packs, and the three streamlets having offset addresses starting at 52K, 56K, and 96K correspond to the claimed “non-sequential” data packs.

Thus, in the proposed combination the “application I/O requests correspond[] to sequential and non-sequential data packs” according to limitation [1.b.iv]. *See* Ex. 1003 ¶¶ 173–175.

(f) Limitation [1.b.v]

Claim 1 recites “said data packs being portions of a complete data stream.” Ex. 1001, 12:51–52.

Petitioner contends that Eylon teaches this limitation because (1) a streamed file in Eylon constitutes “a complete data stream” and

(2) a streamlet in Eylon constitutes a “portion” of “a complete data stream.” Pet. 37 (emphases omitted) (citing Ex. 1003 ¶¶ 187–188; Ex. 1005, 3:45–55, 4:10–50, 6:43–49, code (57), Fig. 7).

Patent Owner “does not dispute that the Eylon streamlets might be portions of a complete data stream.” Resp. 28.

We agree with Petitioner that Eylon teaches limitation [1.b.v]. *See* Pet. 37–38; Ex. 1003 ¶¶ 186–189. Specifically, Eylon explains that an application typically has multiple files, such as application FOOBAR having files FOOBAR.EXE, FOOBAR.DLL, and DEFAULT.DAT. Ex. 1005, 11:57–62, Fig. 7; *see* Ex. 1003 ¶¶ 165–166. Eylon discloses breaking an application’s files into multiple streamlets, i.e., “parts into which the application has been divided.” Ex. 1005, 3:57–59, code (57); *see* Ex. 1003 ¶ 162. Hence, a streamed file in Eylon constitutes “a complete data stream,” and a streamlet in Eylon constitutes a “portion” of “a complete data stream.” *See* Ex. 1003 ¶¶ 187–188; Ex. 1005, 3:45–59, 4:10–50, 6:43–49, 11:57–62, code (57), Fig. 7.

Thus, in the proposed combination the “data packs [are] portions of a complete data stream” according to limitation [1.b.v]. *See* Ex. 1003 ¶¶ 187–188.

(g) Limitation [1.b.vi]

Claim 1 recites “said I/O requests determining an order in which data packs are sought for the application.” Ex. 1001, 12:52–53.

Petitioner contends that Eylon teaches this limitation because Eylon’s streaming application issues I/O requests when it needs certain data, and “therefore the application’s I/O requests determine the order in which the data is sought.” Pet. 38 (citing Ex. 1003 ¶ 191; Ex. 1006 ¶ 33). Petitioner

also contends that Eylon “explains that ‘as the streaming application executes and paging or data I/O requests are generated to retrieve require[d] code or data.’” *Id.* (alteration by Petitioner) (quoting Ex. 1005, 4:17–19).

Patent Owner disputes that Eylon teaches limitation [1.b.vi]. *See* Resp. 28–29. Specifically, Patent Owner asserts that in Eylon’s system a server “performs predictive transmission using a prediction engine on the server.” *Id.* at 29 (citing Ex. 1005, 4:51–62). Patent Owner also asserts that if a client’s file system lacks requested streamlets “the server for transmission dictates a next series of streamlets to transmit.” *Id.* (citing Ex. 1005, 10:13–33).

We agree with Petitioner that Eylon teaches limitation [1.b.vi]. *See* Pet. 38; Ex. 1003 ¶¶ 190–193. As Petitioner contends, Eylon’s streaming application issues I/O requests when it needs certain data, and “therefore the application’s I/O requests determine the order in which the data is sought.” Ex. 1003 ¶ 191; *see* Ex. 1005, 4:17–25; Pet. 38.

Thus, in the proposed combination the “I/O requests determin[e] an order in which data packs are sought for the application” according to limitation [1.b.vi]. *See* Ex. 1003 ¶¶ 190–193.

Patent Owner’s assertion that in Eylon’s system a server “performs predictive transmission using a prediction engine on the server” does not distinguish claim 1 from the references. *See* Resp. 29. As discussed above, in Eylon’s system a server not only pushes data to a client using a predictive streaming engine, the server also services the client’s on-demand requests. Ex. 1005, 7:4–8, 7:56–64, 12:43–51, 14:51–54; *see* Ex. 1035 ¶ 17; *supra* § III.D.3(a).

Additionally, we disagree with Patent Owner’s assertion that if a client’s file system lacks requested streamlets “the server for transmission dictates a next series of streamlets to transmit.” *See* Resp. 29. As discussed above, Eylon discloses that when “the user strays from the predictive path,” the client issues an on-demand request (e.g., a fetch request) to the server, and the server responds to the on-demand request with appropriate streamlets. Ex. 1005, 6:25–27, 6:37–39, 7:4–8; *see id.* at 4:25–28, 9:57–59, 10:53–56, 11:1–10, 13:2–4, 13:35–37; Ex. 1003 ¶¶ 149, 192; *supra* § III.D.3(e).

(h) Limitation [1.c.i]

Claim 1 recites “said on-demand requester object comprising a minifilter associated with a filter manager in an I/O stack.” Ex. 1001, 12:53–55.

Petitioner contends that Christiansen teaches this limitation because Christiansen discloses filter manager 220 in an I/O stack and filters 230, 231, and 232 associated with filter manager 220. *See* Pet. 39–40 (citing Ex. 1003 ¶ 202; Ex. 1006 ¶ 33, Fig. 2). Petitioner also contends that filters 230, 231, and 232 are “minifilters” because “each is managed such that it is called only for operations for which is it registered.” *Id.* at 40 (citing Ex. 1003 ¶ 205; Ex. 1006 ¶¶ 34, 36).

Patent Owner disputes that Christiansen teaches limitation [1.c.i]. *See* Resp. 29–31; Sur-reply 16–17. Specifically, Patent Owner asserts that Christiansen discloses a minifilter residing only “on the server rather than the client.” Resp. 29–30 (citing Ex. 1006 ¶¶ 48, 50–51, Figs. 5–7); *see id.* at 39; Sur-reply 17. Patent Owner also asserts that because “the minifilter resides on the server rather than the client, caching occurs, as illustrated in

Fig. 5, in defined cache drives, 525, 526, and 527, which also reside on the server side of the client/server gulf.” Resp. 30. Patent Owner further asserts that “there is no motivation to combine the references in order to deploy a minifilter to do what Eylon accomplishes on its own.” Sur-reply 17; *see* Resp. 30–31.

We agree with Petitioner that Christiansen teaches limitation [1.c.i]. *See* Pet. 39–40; Ex. 1003 ¶¶ 198–207; Ex. 1035 ¶ 22. Specifically, Christiansen discloses a computer system with an I/O stack including I/O manager 215 and filter manager 220 that service file system 225. Ex. 1006 ¶¶ 10, 32–33, Fig. 2; *see* Ex. 1003 ¶¶ 202–203. An application makes a file system request through an application programming interface to I/O manager 215. Ex. 1006 ¶ 33. After receiving the file system request, I/O manager 215 “determine[s] what I/O request or requests should be issued to fulfill” the file system request and “send[s] each I/O request” to filter manager 220. *Id.*

Additionally, Christiansen discloses that filters 230, 231, and 232 register with filter manager 220, e.g., during an initialization routine. Ex. 1006 ¶¶ 32, 34. Typically, a filter registers only for particular I/O operations, i.e., I/O operations “it may be interested in processing,” such as one or more “create, read, write, close, [and] rename” operations. *Id.* ¶ 34.

Because Christiansen’s filters 230, 231, and 232 “typically will only register for” particular I/O operations with filter manager 220, Christiansen teaches “a minifilter associated with a filter manager in an I/O stack” according to limitation [1.c.i]. *See* Ex. 1003 ¶¶ 72–79, 198–207; Ex. 1026 ¶¶ 13–22; *supra* § III.C.3.

We disagree with Patent Owner’s assertion that Christiansen discloses a minifilter residing only “on the server rather than the client.” *See* Resp. 29–30. Christiansen explains that the invention may operate in “numerous” general-purpose and special-purpose “computing system environments or configurations.” Ex. 1006 ¶ 21; *see* Ex. 1035 ¶ 22. Those environments or configurations include “personal computers, server computers, hand-held or laptop devices, multiprocessor systems, microcontroller-based systems, set top boxes, programmable consumer electronics, [and] network PCs.” Ex. 1006 ¶ 21; *see* Ex. 1035 ¶ 22. An ordinarily skilled artisan would have understood that “personal computers, hand held or laptop devices, set top boxes, consumer electronics, and network PCs act as client devices.” Ex. 1035 ¶ 22.

Additionally, Christiansen describes several embodiments of the invention, e.g., as depicted in Figures 2 through 7. Ex. 1006 ¶¶ 10–15, 32–58, Figs. 2–7. Patent Owner acknowledges that Christiansen describes “distinct embodiments of various features.” Resp. 30.

For the embodiments depicted in Figure 5, 6, and 7, Christiansen discloses a specific computing environment including a server, such as Figure 5’s environment with a file server and content servers containing filters. Ex. 1006 ¶¶ 45–58, Figs. 5–7. For the embodiments depicted in Figures 2, 3, and 4, however, Christiansen discloses a general computing environment in a “machine” with a user mode and a kernel mode. *Id.* ¶¶ 39, 43–44, Figs. 2–4. Christiansen explains that filters may (1) “execute in user mode or in kernel mode” and (2) “perform any set of actions that may be performed by a kernel-mode or user-mode process.” *Id.* ¶¶ 39, 43; *see id.* ¶ 21; Ex. 1035 ¶ 22. Both a server and a client exemplify a “machine” with

a user mode and a kernel mode. *See* Ex. 1006 ¶¶ 20–21, 39, 43–44, 52, Fig. 1. In Figure 5’s environment, for instance, a content server could include a filter for streaming data from the file server at ten units per second, while a client could include a filter for streaming data from the content server at five units per second. *See id.* ¶¶ 13, 45, Fig. 5.

For these reasons, Christiansen discloses that a filter may reside on a server or a client (or both). *See* Ex. 1006 ¶¶ 21, 39, 43; Ex. 1035 ¶ 22. Patent Owner overlooks Christiansen’s disclosure that a filter may reside on a server or a client (or both). *See, e.g.*, Resp. 29–31; Sur-reply 16–17.

As for Patent Owner’s assertion that “there is no motivation to combine the references in order to deploy a minifilter to do what Eylon accomplishes on its own,” we address that assertion below. *See* Sur-reply 17; *infra* § III.D.3(o).

(i) Limitation [1.c.ii]

Claim 1 recites “said filter manager being configured to call the minifilter.” Ex. 1001, 12:55–56.

Petitioner contends that Christiansen teaches this limitation because Christiansen discloses that filter manager 220 may “call and pass the callback data to each filter that has registered to receive callbacks for the type of I/O received by the filter manager.” *See* Pet. 41 (emphasis omitted) (quoting Ex. 1006 ¶ 36).

Patent Owner makes no arguments specific to limitation [1.c.ii]. *See, e.g.*, Resp. 23–36; Sur-reply 13–17. Nonetheless, the burden remains on Petitioner to demonstrate unpatentability. *See Dynamic Drinkware*, 800 F.3d at 1378.

We agree with Petitioner that Christiansen teaches limitation [1.c.ii]. *See* Pet. 41; Ex. 1003 ¶¶ 209–212, 218. Specifically, Christiansen discloses that a filter may register for pre-callbacks or post-callbacks (or both). Ex. 1006 ¶ 35. A “pre-callback is called as data associated with an I/O request propagates from the I/O manager 215 towards the file system 225.” *Id.* A “post-callback is called during the completion of the I/O request as data associated with the I/O request propagates from the file system 225 towards the I/O manager 215.” *Id.* Further, a filter may “specify whether the filter should be notified” for a pre-callback or a post-callback (or both) for each type of I/O operation that the filter has registered with filter manager 220. *Id.*

Christiansen also discloses that filter manager 220 may “create a data structure” for the callback data. Ex. 1006 ¶ 36. Filter manager 220 may “call and pass the callback data to each filter that has registered to receive callbacks for the type of I/O received by the filter manager.” *Id.*; *see id.* ¶ 37. Filter manager 220 passes the callback data “associated with a particular type of I/O request to each registered filter sequentially in an order in which the registered filters are ordered.” *Id.* ¶ 37. For example, if filter 230 precedes filter 232 in processing I/O requests and if filter manager 220 receives a read I/O request, “the filter manager 220 may first call and pass the callback data to the filter 230 and after the filter 230 has processed the callback data, the filter manager 220 may then call and pass the callback data (as modified, if at all) to the filter 232.” *Id.*

Thus, in the proposed combination a filter manager would have been “configured to call the minifilter” according to limitation [1.c.ii]. *See* Ex. 1003 ¶¶ 209–212, 218.

(j) Limitation [1.c.iii]

Claim 1 recites “said minifilter having a preoperation callback routine for the I/O request registered with the filter manager.” Ex. 1001, 12:56–58.

Petitioner contends that Christiansen teaches this limitation because Christiansen discloses that a filter may “specify whether the filter should be notified for pre-callbacks and post callbacks” for each type of I/O operation that the filter has registered with the filter manager. *See* Pet. 42 (emphasis omitted) (quoting Ex. 1006 ¶ 35). Petitioner also contends that the computer code in a filter for processing a pre-callback constitutes “a preoperation callback routine.” *Id.* (citing Ex. 1003 ¶ 219).

Patent Owner disputes that Christiansen teaches limitation [1.c.iii]. *See* Resp. 32. Specifically, Patent Owner asserts that “there is no reason for” an ordinarily skilled artisan “to create any specific preoperation callback routine.” *Id.* Patent Owner also asserts that Petitioner identifies “no structure” permitting “a client-resident detector [to] interact with a server-resident minifilter.” *Id.*

Petitioner responds to Patent Owner’s assertion that Petitioner identifies “no structure” permitting “a client-resident detector [to] interact with a server-resident minifilter” by contending that Patent Owner “does not explain how that assertion relates to” limitation [1.c.iii]. Reply 14.

We agree with Petitioner that Christiansen teaches limitation [1.c.iii]. *See* Pet. 41–42; Ex. 1003 ¶¶ 213–220. Specifically, Christiansen discloses that a filter may “specify whether the filter should be notified” for a pre-callback or a post-callback (or both) for each type of I/O operation that the filter has registered with filter manager 220. Ex. 1006 ¶ 35; *see* Ex. 1003 ¶ 218. As Petitioner contends, the computer code in a filter for processing

a pre-callback constitutes “a preoperation callback routine.” *See* Ex. 1003 ¶¶ 218–219; Pet. 42.

Thus, in the proposed combination a managed filter would have had “a preoperation callback routine for the I/O request registered with the filter manager” according to limitation [1.c.iii]. *See* Ex. 1003 ¶¶ 213–220.

Patent Owner’s assertion that “there is no reason for” an ordinarily skilled artisan “to create any specific preoperation callback routine” identifies no deficiency in Christiansen’s disclosure relative to limitation [1.c.iii]. *See* Resp. 32. Moreover, claim 1 does not require that the “preoperation callback routine” perform a specific function. *See* Ex. 1001, 12:44–67.

As for Patent Owner’s assertion that Petitioner identifies “no structure” permitting “a client-resident detector [to] interact with a server-resident minifilter,” we agree with Petitioner that Patent Owner “does not explain how that assertion relates to” limitation [1.c.iii]. *See* Resp. 31–32; Reply 14; Sur-reply 13–17. Additionally, Christiansen’s disclosure of a managed filter including a pre-callback satisfies any structural requirement in limitation [1.c.iii]. *See* Ex. 1003 ¶¶ 215–219; Ex. 1006 ¶ 35.

(k) Limitation [1.c.iv]

Claim 1 recites “said minifilter being configured to receive each I/O request from the application.” Ex. 1001, 12:58–59.

Petitioner contends that the combined disclosures in Christiansen and Eylon teach this limitation. *See* Pet. 42–43. Specifically, Petitioner asserts that in the proposed combination Eylon’s streaming functionality, e.g., streaming support system 102, would have been implemented as a managed filter in Christiansen’s system, e.g., as one of filters 230, 231, and 232 in

Christiansen's Figure 2. *Id.* at 42 (citing Ex. 1003 ¶ 224; Ex. 1005, 4:10–22); *see id.* at 24–28; Reply 20 & n.12. Petitioner also asserts that in the proposed combination I/O requests would have been directed to the managed filter with streaming functionality. Pet. 42–43 (citing Ex. 1003 ¶ 225).

To support Petitioner's assertions, Dr. Houh testifies that an ordinarily skilled artisan would have been motivated to implement Eylon's streaming functionality, e.g., streaming support system 102, as a managed filter in Christiansen's system because a managed filter would have provided "numerous advantages." Ex. 1003 ¶¶ 127–128, 132–133, 140. Dr. Houh also testifies that "having a single minifilter handle each I/O request from the specific streaming application disclosed in Eylon would have been an efficient programming technique." *Id.* ¶ 226 (emphases omitted).

Patent Owner disputes that the combined disclosures in Christiansen and Eylon teach limitation [1.c.iv]. *See* Resp. 32–33. Specifically, Patent Owner asserts that "the minifilter must reside on the server to observe changes imposed on the remote files, i.e. there is an [sic] minifilter element that resides on the server to cooperate with the client's communication module to prevent free access to the server." *Id.* at 33.

We agree with Petitioner that the combined disclosures in Christiansen and Eylon teach limitation [1.c.iv]. *See* Pet. 42–43; Ex. 1003 ¶¶ 221–227. Specifically, Eylon discloses streaming support system 102 that receives each I/O request from a streaming application. Ex. 1005, 4:1–30, 5:30–34, 7:56–8:41, 10:34–46, Figs. 3–4, 6A; *see* Ex. 1003 ¶¶ 120–121, 155–161. Christiansen discloses a managed filter corresponding to the claimed "minifilter." Ex. 1006 ¶¶ 10, 32–37, Fig. 2;

see Ex. 1003 ¶¶ 111–114, 198–207. As Dr. Houh testifies, an ordinarily skilled artisan would have been motivated to implement Eylon’s streaming functionality, e.g., streaming support system 102, as a managed filter in Christiansen’s system. Ex. 1003 ¶¶ 132–133, 140; *see infra* § III.D.3(o). As Dr. Houh also testifies, “having a single minifilter handle each I/O request from the specific streaming application disclosed in Eylon would have been an efficient programming technique.” Ex. 1003 ¶ 226 (emphases omitted).

Thus, in the proposed combination a managed filter would have been “configured to receive each I/O request from the application” according to limitation [1.c.iv]. *See* Ex. 1003 ¶¶ 225–226.

Patent Owner’s assertion that “the minifilter must reside on the server” to “cooperate with the client’s communication module to prevent free access to the server” does not address the proposed combination. *See* Pet. 24–25; Resp. 33. Eylon discloses that “there is preferably no direct access from the operating system level to the server” to secure “the streamed application from unauthorized access and use.” Ex. 1005, 14:1–13. But Petitioner does not propose including that optional security functionality in the proposed combination. *See* Pet. 24–25; Reply 20 & n.12; Ex. 1003 ¶¶ 127–128.

Additionally, as discussed above, Eylon discloses streaming support system 102 installed on a client device, e.g., a personal computer, and Christiansen discloses that a filter may reside on a client device, e.g., a personal computer. Ex. 1005, 7:56–8:4, 8:39–41, 9:4–7, Figs. 3–4; Ex. 1006 ¶¶ 21, 39, 43; *see* Ex. 1003 ¶¶ 152–159; Ex. 1005, 4:1–36, 8:18–19; Ex. 1035 ¶ 22; *supra* §§ III.D.3(b), III.D.3(h). For this reason too,

we disagree with Patent Owner's assertion that "the minifilter must reside on the server." *See* Resp. 33.

(l) Limitation [1.c.v]

Claim 1 recites "said minifilter being further configured to reference a table that includes names and paths with at least one offset address and length where each data pack required to fulfill each I/O request is located." Ex. 1001, 12:59–63.

Petitioner contends that Eylon teaches this limitation because Eylon's VFS 160 includes file structure library 210, streamlet map 220, and streamlet library 230 where file structure library 210 and streamlet map 220 collectively include "names and paths with at least one offset address and length where each data pack required to fulfill each I/O request is located." *See* Pet. 43–45; Reply 15. Specifically, Petitioner asserts that file structure library 210 contains "the complete folder/file structure for each streaming application," including file names and paths. Pet. 43 (emphasis omitted) (quoting Ex. 1005, 11:38–40); *see* Reply 15. Petitioner asserts that streamlet map 220 associates "the various streamlets in the library 230 with a respective file name and position within that file" and contains file names, offset addresses, and streamlet lengths. Pet. 43–44 (quoting Ex. 1005, 11:42–44). Petitioner explains that the differences between the offset addresses in streamlet map 220 indicate "the length (size) of each streamlet." *Id.* at 44 (citing Ex. 1005, 11:56–12:15, Figs. 7–8). Petitioner further asserts that in the proposed combination Eylon's streaming functionality would have been implemented as a managed filter in Christiansen's system where the information in file structure library 210 and streamlet map 220 "is

included in the mapping table of Christiansen.” *Id.* at 45 (emphasis omitted) (citing Ex. 1003 ¶¶ 233–234).

In addition, Petitioner contends that for Eylon’s streaming support system 102 “to be responsive to each I/O request from the application and to locate the required streamlets, it would need to know the ‘names and paths with at least one offset address and length where each data pack required to fulfill each I/O request is located.’” Pet. 45–46 (emphasis omitted). Petitioner also contends that an ordinarily skilled artisan would have been motivated to “include such information in the mapping table of the combined system” to “permit the system to be responsive, in an efficient manner, to each I/O request from the application.” *Id.* at 46.

Patent Owner disputes that Eylon teaches limitation [1.c.v]. *See* Resp. 34–35; Sur-reply 17. Specifically, Patent Owner asserts that streamlet map 220 “is little more than instructions to reconstruct files from streamlets but does not say where to find those streamlets.” Resp. 34. Patent Owner asserts that the absence “rather than presence of the data” in streamlet map 220 triggers a “fetch request to the server.” *Id.* (citing Ex. 1005, 12:17–22); *see id.* at 37. According to Patent Owner, no part of file structure library 210 or streamlet map 220 “looks outside of the client ‘where each data pack required to fulfill each I/O request is located’” as required by limitation [1.c.v]. *Id.* at 34. Patent Owner also asserts that “Eylon does not teach file lengths.” Sur-reply 17 (citing Ex. 2134, 123:1–125:25).

We agree with Petitioner that Eylon teaches limitation [1.c.v]. *See* Pet. 43–47; Ex. 1003 ¶¶ 228–238. Specifically, Eylon discloses that VFS 160 includes VFS manager 200 and the following data or library files that “contain the streamed data and information needed to map that data” to

each streaming application: file structure library 210, streamlet map 220, and streamlet library 230. Ex. 1005, 11:34–44, 11:64–67, Fig. 7. File structure library 210 contains “the complete folder/file structure for each streaming application.” *Id.* at 11:38–40, 11:65–67, Fig. 7. Streamlet library 230 contains the streamlets received from the server. *Id.* at 11:40–42, 12:32–37, Fig. 7. Streamlet map 220 associates “the various streamlets in the library 230 with a respective file name and position within that file.” *Id.* at 11:42–44, 12:9–13, Fig. 7. Thus, VFS 160 “knows the framework, files, folder/directory structure, etc.” of each streaming application. *Id.* at 12:52–57.

In addition, Eylon discloses an example where file structure library 210 includes the following file names and paths:

- (1) file FOOBAR.EXE at path /FOOBAR/FOOBAR.EXE;
- (2) file FOOBAR.DLL at path /FOOBAR/FOOBAR.DLL;
and
- (3) file DEFAULT.DAT at path
/FOOBAR/DATA/DEFAULT.DAT.

Ex. 1005, 11:57–62, Fig. 7. In that example, streamlet map 220 associates six named streamlets comprising part of file FOOBAR.EXE with six offset addresses, i.e., the offsets of 0K, 4K, 8K, 52K, 56K, and 96K. *Id.* at 12:5–16, Figs. 7–8. Also, streamlet map 220 associates one named streamlet comprising part of file FOOBAR.DLL with one offset address, i.e., an offset of 0K following the offset of 96K for file FOOBAR.EXE. *Id.* at 12:5–9, 12:43–46, Figs. 7–8.

In streamlet map 220, the differences between the offset addresses determine a “length” for each streamlet. Ex. 1003 ¶¶ 230–231; *see* Ex. 1005, 12:4–16, Figs. 7–8. As an example, Eylon’s Figure 7 shows

streamlet FOOBAR001 starting at an offset address of 4K and streamlet FOOBAR002 starting at an offset address of 8K such that streamlet FOOBAR001 has a length of 4K. Ex. 1005, 12:11–13, Fig. 7; *see* Ex. 1003 ¶ 231. Consistent with this, the ’808 patent explains that “start and end offsets” may serve as a “size indicator.” Ex. 1001, 6:44–47.

Together, file structure library 210 and streamlet map 220 include file names and paths with offset addresses and lengths where streamlets required to fulfill I/O requests are located. *See* Ex. 1003 ¶¶ 230–232; Ex. 1005, 11:34–44, 11:57–62, 12:9–16, 12:43–46, Figs. 7–8. Thus, file structure library 210 and streamlet map 220 correspond to the claimed “table” containing location information. *See* Ex. 1003 ¶¶ 230–232. Consistent with this, the ’808 patent explains that a “reference table is not limited to a table structure, or even a single table,” and that “one or more local and/or remote reference sources may comprise a reference table.” Ex. 1001, 6:49–52.

As the excerpt from Eylon’s Figure 7 reproduced below shows, however, streamlet map 220 does not contain location information for “each data pack required to fulfill each I/O request” because streamlet map 220 contains location information for locally stored streamlets but not remotely stored streamlets:

<u>FILENAME</u>	<u>OFFSET</u>	STREAMLET
		<u>ID</u>
FOOBAR.EXE	0K	FOOBAR000
FOOBAR.EXE	4K	FOOBAR001
FOOBAR.EXE	8K	FOOBAR002
FOOBAR.EXE	12K	X
FOOBAR.EXE	16K	X
. . .		
FOOBAR.EXE	48K	X
FOOBAR.EXE	52K	FOOBAR003
FOOBAR.EXE	56K	FOOBAR004
FOOBAR.EXE	60K	X
. . .		
FOOBAR.EXE	96K	FOOBAR005
FOOBAR.DLL	0	FOOBAR006
FOOBAR.DLL	4K	x
. . .		
FOOBAR.DLL	32K	x

The above excerpt from Eylon’s Figure 7 depicts six named streamlets comprising part of file FOOBAR.EXE starting at offset addresses of 0K, 4K, 8K, 52K, 56K, and 96K, respectively, and one named streamlet comprising part of file FOOBAR.DLL starting at an offset address of 0K following the offset address of 96K for file FOOBAR.EXE. *See* Ex. 1005, 12:5–15, Fig. 7. These streamlets are stored in streamlet library 230. *See id.* at 11:40–42, 12:32–37, Fig. 7. Thus, streamlet map 220 contains location information for locally stored streamlets but not remotely stored streamlets. *Id.* at Fig. 7.

Although streamlet map 220 does not contain location information for “each data pack required to fulfill each I/O request,” Eylon indicates that streaming support system 102 contains the location information not included in streamlet map 220. In particular, Eylon explains that streaming support system 102 may issue fetch requests specifically identifying remotely stored streamlets, e.g., by specifying “the file name, offset, and length.” Ex. 1005, 11:1–10. To issue fetch requests specifically identifying remotely stored

streamlets, streaming support system 102 must contain location information for the remotely stored streamlets as well as the location information included in streamlet map 220. *See id.* at 11:1–10.

Moreover, an ordinarily skilled artisan would have been motivated to modify streamlet map 220 to include location information for “each data pack required to fulfill each I/O request.” *See* Ex. 1003 ¶¶ 226, 235–239. As Dr. Houh testifies, an ordinarily skilled artisan would have been motivated to “include such information in the mapping table of the combined system” to “permit the system to be responsive, in an efficient manner, to each I/O request from the application.” *Id.* ¶ 235; *see id.* ¶ 226. As Petitioner notes, Patent Owner does not contest this motivation to modify streamlet map 220. *See* Resp. 34–35; Reply 16; Sur-reply 17.

Thus, in the proposed combination a managed filter would have been “further configured to reference a table that includes names and paths with at least one offset address and length where each data pack required to fulfill each I/O request is located” according to limitation [1.c.v]. *See* Ex. 1003 ¶¶ 228–238.

Patent Owner’s assertion that the absence “rather than presence of the data” in streamlet map 220 triggers a “fetch request to the server” does not fully reflect how Eylon’s system operates. As discussed above, streaming support system 102 does not trigger a “fetch request to the server” if the client contains the requested streamlets. *See, e.g.,* Ex. 1003 ¶ 124; Ex. 1005, 4:10–25; *supra* § III.D.3(d). If the client lacks the requested streamlets, however, “a fetch request is issued to the server for the appropriate streamlet blocks.” Ex. 1005, 4:25–28; *see id.* at 10:53–56, 11:1–10, 13:2–4,

15:28–32, 17:1–7, 17:56–59, code (57), Fig. 6A; Ex. 1003 ¶¶ 124, 168; Ex. 1035 ¶ 20.

As for Patent Owner’s assertion that “Eylon does not teach file lengths,” limitation [1.c.v] does not require “file lengths.” *See* Ex. 1001, 12:59–63; Sur-reply 17. Rather, this limitation requires a “length” for each “data pack.” Ex. 1001, 12:59–63. In Eylon’s streamlet map 220, the differences between the offset addresses determine a “length” for each streamlet, and thus determine a “length” for each “data pack.” Ex. 1003 ¶¶ 230–231; *see* Ex. 1005, 12:4–16, Figs. 7–8.

(m) Limitation [1.c.vi]

Claim 1 recites “said minifilter being further configured to determine if the data pack has been streamed to the system.” Ex. 1001, 12:63–64.

Petitioner contends that Eylon teaches this limitation because Eylon discloses determining if “the requested streamlets are present” in VFS 160 when a streaming application executes. *See* Pet. 47 (emphasis omitted) (quoting Ex. 1005, 4:23–25). Petitioner also contends that Eylon’s system “performs a similar check for whether the streamlet has already been streamed to the system when first starting an application.” *Id.* (citing Ex. 1005, 9:50–54).

Patent Owner disputes that Eylon teaches limitation [1.c.vi]. *See* Resp. 35–36. Specifically, Patent Owner asserts that this limitation “requires that a page execution fault be generated by the ***absence of the data in the VFS***,” whereas Eylon discloses determining if “the requested streamlets ***are present in the VFS***” when a streaming application executes. *Id.* at 35 (emphases by Patent Owner).

We agree with Petitioner that Eylon teaches limitation [1.c.vi]. *See* Pet. 47; Ex. 1003 ¶¶ 240–249. Specifically, Eylon discloses determining if “the requested streamlets are present” in VFS 160 when a streaming application executes. Ex. 1005, 4:23–25; *see id.* at 4:10–22, 9:50–61, 10:34–56, 15:22–27, 16:59–65, 17:47–53, code (57), Fig. 6A. In particular, FSD 150 receives a request to retrieve required code or data and passes the request to VFS 160 to determine “if the desired data is available” in VFS 160. *Id.* at 10:43–46, Fig. 6A; *see id.* at 4:17–22.

Thus, in the proposed combination a managed filter would have been “further configured to determine if the data pack has been streamed to the system” according to limitation [1.c.vi]. *See* Ex. 1003 ¶¶ 240–249.

We disagree with Patent Owner’s assert that limitation [1.c.vi] “requires that a page execution fault be generated by the absence of the data in the VFS.” *See* Resp. 35 (emphasis omitted). Nothing in the ’808 patent’s claim language, specification, or prosecution history “requires that a page execution fault be generated” for a minifilter to “determine if the data pack has been streamed to the system” according to limitation [1.c.vi]. *See, e.g.,* Ex. 1001, 2:41–53, 7:23–60, 12:44–67, code (57), Fig. 4; Ex. 1002, 61–66, 76–84, 101–06.

(n) Limitation [1.d]

Claim 1 recites “said application being a program configured to use the data packs that fulfill each I/O request when the data packs are available on the computing device.” Ex. 1001, 12:64–67.

Petitioner contends that Eylon teaches this limitation because Eylon discloses that (1) if “the requested streamlets are present” in VFS 160, “the data is returned to the operating system and the streaming program continues

normal operation,” and (2) when “the streamlets are returned [from a server], they are stored in the VFS and the read or paging request from the operating system is satisfied.” Pet. 48 (alteration by Petitioner) (emphases omitted) (quoting Ex. 1005, 4:23–25, 4:28–30).

Patent Owner “does not contest that when all the data is on the client machine, the client computer can use it—no streaming is necessary nor would it be requested.” Resp. 36.

We agree with Petitioner that Eylon teaches limitation [1.d]. See Pet. 47; Ex. 1003 ¶¶ 250–256. As discussed above, Eylon discloses determining if “the requested streamlets are present” in VFS 160 when a streaming application executes. Ex. 1005, 4:23–25; *see id.* at 4:10–22, 9:50–61, 10:34–56, 15:22–27, 16:59–65, 17:47–53, code (57), Fig. 6A; *supra* § III.D.3(m). If “the requested streamlets are present” in VFS 160, “the data is returned to the operating system and the streaming program continues normal operation.” Ex. 1005, 4:23–25; *see id.* at 10:46–47, 16:66–67, 17:47–55, Fig. 6A; Ex. 1003 ¶¶ 124, 168; *supra* §§ III.D.3(d), III.D.3(l). But if “one or more of the requested streamlet blocks are absent” from VFS 160, “a fetch request is issued to the server for the appropriate streamlet blocks.” Ex. 1005, 4:25–28; *see id.* at 10:53–56, 11:1–10, 13:2–4, 15:28–32, 17:1–7, 17:56–59, code (57), Fig. 6A; Ex. 1003 ¶¶ 124, 168; Ex. 1035 ¶ 20; *supra* § III.D.3(l). VFS 160 stores the streamlet blocks received from the server for use by the streaming application. Ex. 1005, 4:28–36, 11:40–42, 11:62–67, 12:32–42, Fig. 7.

Thus, in the proposed combination Eylon’s streaming application would have been “configured to use the data packs that fulfill each I/O

request when the data packs are available on the computing device” according to limitation [1.d]. *See* Ex. 1003 ¶¶ 250–256.

(o) Alleged Reasons for Combining the Teachings of the References

Petitioner contends that an ordinarily skilled artisan would have been motivated to combine Eylon’s teachings with Christiansen’s teachings to implement Eylon’s streaming functionality as a managed filter in Christiansen’s system for several reasons. *See* Pet. 24–28; Reply 20 & n.12. Specifically, Petitioner asserts that implementing Eylon’s streaming functionality as a managed filter in Christiansen’s system would have provided “numerous advantages over legacy filters.” Pet. 26–27 (citing Ex. 1003 ¶¶ 132–133; Ex. 1010, 3). Petitioner also asserts that Christiansen identifies “a key benefit of using managed filters by stating that ‘[f]or efficiency, each filter typically will only register for file system requests in which it may be interested in processing.’” *Id.* at 28 (alteration by Petitioner) (emphasis omitted) (quoting Ex. 1006 ¶ 34).

To support Petitioner’s assertions, Dr. Houh testifies that an ordinarily skilled artisan would have been motivated to implement Eylon’s streaming functionality as a managed filter in Christiansen’s system because “it was known in the prior art that employing a managed filter” provided “numerous advantages.” Ex. 1003 ¶¶ 127–128, 132–133. He identifies the following as known advantages of employing a managed filter:

- “a simpler, more reliable filter driver with less development effort”;
- “dynamic load and unload, attach and detach of filters”;
- “the ability to attach filters at a well-defined location in the filter stack”;

- “better context management: fast, clean, reliable contexts for file objects, streams, files, instances, and volumes”;
- “a host of utility routines including support for looking up names and caching them for efficient access, communication between minifilters and user mode services, and IO queuing”; and
- “support for non-recursive I/O so managed filter generated I/O can easily be seen only by lower filters and the file system.”

Id. ¶ 132 (citing Ex. 1010, 3); *see* Ex. 2134, 109:20–110:17, 112:21–114:5; Pet. 26–27. During his deposition, Dr. Houh described the ability to dynamically load and unload as “a huge advantage, and one that would have been recognized by a person of ordinary skill.” Ex. 2134, 114:2–5.

Additionally, Dr. Houh testifies that an ordinarily skilled artisan “would have been further motivated to make this combination in order to take advantage of Christiansen’s more efficient filter driver system” because Christiansen “discloses that ‘[f]or efficiency, each filter typically will only register for file system requests in which it may be interested in processing.’” Ex. 1003 ¶ 140 (alteration by Dr. Houh) (emphases omitted) (quoting Ex. 1006 ¶ 34). He explains that an ordinarily skilled artisan “would have understood that this architecture permitted filter drivers that had nothing to do with a request to be bypassed, thus making the stack more efficient when compared against the older linear, model of passing a request to each filter in the stack as it travelled downward.” *Id.* ¶ 141 (citing Ex. 1026 ¶ 19). He also explains that “such efficiency would be particularly important” to Eylon’s streaming application because “any substantial latency in the fulfillment of an I/O request could be perceived by the user as poor performance or even an annoying trait of the application.” *Id.*

Petitioner also contends that the proposed combination “would have been only the arrangement of old elements” and would have yielded “no more” than an ordinarily skilled artisan “would expect from such an arrangement.” Pet. 26 (citing Ex. 1003 ¶ 131). To support Petitioner’s contention, Dr. Houh testifies that an ordinarily skilled artisan would have had a reasonable expectation of success implementing Eylon’s streaming functionality as a managed filter in Christiansen’s system because the proposed combination “would have been only the arrangement of old elements” and would have yielded “no more” than an ordinarily skilled artisan “would expect from such an arrangement.” Ex. 1003 ¶ 131.

Patent Owner disputes that an ordinarily skilled artisan would have been motivated to combine Eylon’s teachings with Christiansen’s teachings as Petitioner proposes. *See* Resp. 20–24, 32, 51–52; Sur-reply 9–13. Specifically, Patent Owner asserts that Petitioner offers no evidence that Eylon’s streaming support system suffers from “any problem known to” an ordinarily skilled artisan at the time of the invention. Resp. 21. Patent Owner similarly asserts that Eylon has “no specific need” for Christiansen’s managed filter that “might motivate” an ordinarily skilled artisan to combine the teachings of the references. *Id.* at 20–21. Patent Owner also asserts that the proposed combination rests on “impermissible hindsight.” *Id.* at 21; *see id.* at 23, 32, 40, 52; Sur-reply 11, 13.

In addition, Patent Owner contends that Petitioner identifies “only the general advantages of a managed filter system over a legacy filter system.” Sur-reply 10. Patent Owner also contends that Petitioner fails to identify a “specific improvement” in “making the move from a virtual drive to a minifilter architecture.” Resp. 20.

For the reasons stated by Petitioner and supported by Dr. Houh’s testimony, we agree with Petitioner that an ordinarily skilled artisan would have had sound reasons to combine Eylon’s teachings with Christiansen’s teachings as Petitioner proposes. *See* Pet. 26–30; Ex. 1003 ¶¶ 129–146; Ex. 1006 ¶ 34; Ex. 1010, 3; Ex. 2134, 109:20–110:17, 112:21–114:5. We also agree that an ordinarily skilled artisan would have had a reasonable expectation of success implementing the proposed combination. *See* Pet. 26; Ex. 1003 ¶ 131.

As Dr. Houh testifies, an ordinarily skilled artisan would have been motivated to implement Eylon’s streaming functionality as a managed filter in Christiansen’s system because “it was known in the prior art that employing a managed filter” provided “numerous advantages.” Ex. 1003 ¶¶ 127–128, 132–133; *see* Ex. 1010, 3; Ex. 1011, 2–6; Ex. 1013, 7–8. Among other things, a managed filter would have (1) provided a “[s]impler” and “more reliable filter driver with less development effort” and (2) filtered “only operations of interest” unlike “the legacy model where a filter has to hook every entry point to pass through operation.” Ex. 1010, 3; *see* Ex. 1003 ¶ 132; Ex. 2134, 86:9–87:23.

Additionally, Christiansen discloses that “[f]or efficiency, each filter typically will only register for file system requests in which it may be interested in processing.” Ex. 1006 ¶ 34. That disclosure would have motivated an ordinarily skilled artisan to implement Eylon’s streaming functionality as a managed filter to efficiently handle I/O requests and avoid “any substantial latency” that “could be perceived by the user as poor performance or even an annoying trait of the application.” Ex. 1003 ¶¶ 140–141.

Petitioner identifies improved efficiency as a “key benefit” of using Christiansen’s managed filter. Pet. 28. Thus, we disagree with Patent Owner’s contentions that Petitioner identifies “only the general advantages of a managed filter” and fails to identify a “specific improvement” in “making the move from a virtual drive to a minifilter architecture.” *See* Resp. 20; Sur-reply 10. Moreover, Patent Owner mischaracterizes the proposed combination because it includes more than just a virtual drive from Eylon. *See* Pet. 24–25; Resp. 20; Ex. 1003 ¶¶ 127–128. The proposed combination implements Eylon’s streaming functionality as a managed filter in Christiansen’s system. *See* Pet. 24–25; Reply 20 & n.12; Ex. 1003 ¶¶ 127–128. As Petitioner explains, that streaming functionality includes (1) determining “whether requested streamlets are stored locally,” (2) identifying “local or remote addresses for requested streamlets from Christiansen’s mapping table,” (3) “fetching and returning [the] requested streamlets,” and (4) bookkeeping “required to implement such functionality” using Eylon’s file structure library 210 and streamlet map 220. Pet. 25 (emphasis omitted); *see* Ex. 1003 ¶¶ 127–128, 142–145.

Patent Owner’s assertion that Petitioner offers no evidence that Eylon’s streaming support system suffers from “any problem known to” an ordinarily skilled artisan at the time of the invention does not address Petitioner’s position. *See* Pet. 26–28; Resp. 21. Petitioner does not rely on a problem in Eylon’s streaming support system as a reason to combine Eylon’s teachings with Christiansen’s teachings. *See* Pet. 26–28; Ex. 1003 ¶¶ 132–141. Rather, Petitioner relies on a benefit obtained with Christiansen’s teachings. *See* Pet. 26–28; Ex. 1003 ¶¶ 132–141.

Specifically, Eylon’s streaming support system corresponds to a legacy filter that would have benefited from Christiansen’s managed-filter architecture. *See* Ex. 1003 ¶¶ 132–133, 140–141; Ex. 1005, 4:1–30, Fig. 4; Ex. 1006 ¶ 34; Ex. 1010, 3 (identifying advantages of a managed-filter architecture); Ex. 1011, 2–6; Ex. 1013, 7–8; Ex. 2134, 110:18–111:8, 111:19–112:7, 114:9–115:12. The absence of the word “filter” in Eylon does not matter. Eylon’s streaming support system performs a filtering operation by determining whether a requested streamlet is available locally or remotely. Ex. 1005, 4:1–30, Fig. 4; *see* Ex. 1003 ¶ 297; Ex. 1010, 3 (explaining that a filter “performs some type of filtering operation on the file system”). Eylon’s streaming support system operates like Christiansen’s filter 610 to determine whether a requested object is available locally or remotely. *See* Ex. 1006 ¶ 58. And Eylon’s streaming support system operates like Pudipeddi’s hierarchical-storage-management (HSM) filter to manage information retrieval. *See* Ex. 1003 ¶ 273; Ex. 1007, 11:39–42; Ex. 1025, 1; Ex. 1028, 1:13–24.

As for Patent Owner’s assertion that the proposed combination rests on “impermissible hindsight,” an explicit analysis “whether there was an apparent reason to combine” the references reduces the risk of “falling prey to hindsight bias.” *See KSR*, 550 U.S. at 418, 421; Resp. 21. Here, Dr. Houh’s explicit analysis of reasons to combine the references demonstrates that he did not fall prey to hindsight bias. *See* Ex. 1003 ¶¶ 132–141.

Patent Owner contends that “it was the [inventor] who first defined the problem of providing an on-demand streaming object that resides entirely on the client device exploiting a minifilter architecture.” Resp. 23.

But claim 1 does not require “an on-demand streaming object that resides entirely on the client device.” *See* Ex. 1001, 12:44–67. Rather, claim 1 broadly recites installation on “a computing device.” *See id.* at 12:44–45, 12:67.

Additionally, the “law does not require that the references be combined for the reasons contemplated by the inventor.” *In re Beattie*, 974 F.2d 1309, 1312 (Fed. Cir. 1992); *see Outdry Techs. Corp. v. Geox S.p.A.*, 859 F.3d 1364, 1371 (Fed. Cir. 2017). Any “need or problem known in the field of endeavor at the time of invention and addressed by the patent can provide a reason for combining” references. *KSR*, 550 U.S. at 420. Thus, the motivation to combine may come “from the prior art reference itself, from the knowledge of one of ordinary skill in the art, or from the nature of the problem to be solved.” *SIBIA Neurosciences, Inc. v. Cadus Pharm. Corp.*, 225 F.3d 1349, 1356 (Fed. Cir. 2000) (citations omitted).

For example, “an implicit motivation to combine” may result from a desire to make a product or process “stronger, cheaper, cleaner, faster, lighter, smaller, more durable, or more efficient.” *DyStar Textilfarben GmbH v. C.H. Patrick Co.*, 464 F.3d 1356, 1368 (Fed. Cir. 2006). Here, implementing Eylon’s streaming functionality as a managed filter in Christiansen’s system would have made streaming more efficient. *See* Ex. 1003 ¶¶ 140–141; Ex. 1006 ¶ 34.

Patent Owner contends that Eylon’s caching system conflicts with Christiansen’s system because Christiansen’s system transfers and stores entire files according to a “hierarchical file structure.” Resp. 17 (citing Ex. 1006 ¶¶ 46–47). We disagree that Eylon’s caching system conflicts with Christiansen’s system. “Combining the *teachings* of references does not

involve an ability to combine their specific structures.” *In re Nievelt*, 482 F.2d 965, 968 (CCPA 1973) (emphasis by the court). Eylon teaches dividing a file into multiple parts for streaming along with appropriate bookkeeping to track the multiple parts. Ex. 1005, 3:57–63, 11:38–12:30, code (57), Fig. 7; *see id.* at 11:2–7, 12:57–65; Ex. 1003 ¶¶ 127, 162. Patent Owner identifies no technical impediment in applying those teachings to Christiansen’s system. *See, e.g.*, Resp. 17.

Patent Owner characterizes Dr. Houh’s testimony as uncorroborated and conclusory. *See* Resp. 22. We disagree with that characterization. To support his testimony, Dr. Houh cites Christiansen and a 2004 Filter Driver Development Guide as identifying various advantages of a managed filter. Ex. 1003 ¶¶ 132, 140 (citing Ex. 1006 ¶ 34; Ex. 1010, 3). That evidence corroborates Dr. Houh’s testimony. *See* Ex. 1006 ¶ 34; Ex. 1010, 3. Additionally, Dr. Houh provides a detailed explanation why an ordinarily skilled artisan would have had sound reasons to combine Eylon’s teachings with Christiansen’s teachings as Petitioner proposes. *See* Ex. 1003 ¶¶ 129–146.

(p) Alleged Objective Indicia of Nonobviousness

(i) Background

Before reaching a conclusion about obviousness, we consider evidence concerning objective indicia of nonobviousness. *See Apple*, 839 F.3d at 1048. For such evidence to have substantial weight, “its proponent must establish a nexus between the evidence and the merits of the claimed invention.” *ClassCo, Inc. v. Apple, Inc.*, 838 F.3d 1214, 1220 (Fed. Cir. 2016). “[T]here is no nexus unless the evidence presented is ‘reasonably commensurate with the scope of the claims.’” *Id.* (quoting

Rambus Inc. v. Rea, 731 F.3d 1248, 1257 (Fed. Cir. 2013)). The patentee “bears the burden of showing that a nexus exists.” *WMS Gaming, Inc. v. Int’l Game Tech.*, 184 F.3d 1339, 1359 (Fed. Cir. 1999).

A rebuttable presumption of nexus arises “when the patentee shows that the asserted objective evidence is tied to a specific product and that product ‘embodies the claimed features, and is coextensive with them.’” *Fox Factory, Inc. v. SRAM, LLC*, 944 F.3d 1366, 1373 (Fed. Cir. 2019) (quoting *Polaris Indus., Inc. v. Arctic Cat, Inc.*, 882 F.3d 1056, 1072 (Fed. Cir. 2018)). Whether a rebuttable presumption of nexus arises “turns on the nature of the claims and the specific facts.” *Teva Pharm. Int’l GmbH v. Eli Lilly & Co.*, 8 F.4th 1349, 1362 (Fed. Cir. 2021). The presumption analysis should consider the unclaimed features in the product tied to the objective evidence to assess their significance and impact on the correspondence between the patented invention and the product. *Quanergy Sys., Inc. v. Velodyne Lidar USA, Inc.*, 24 F.4th 1406, 1418 (Fed. Cir. 2022). When, “for example, the patented invention is only a small component of the product tied to the objective evidence, there is no presumption of nexus.” *Henny Penny Corp. v. Frymaster LLC*, 938 F.3d 1324, 1333 (Fed. Cir. 2019). Absent a presumption of nexus, the patentee may prove nexus by showing that the objective evidence resulted directly from “the unique characteristics of the claimed invention.” *Fox Factory*, 944 F.3d at 1373–74; see *In re Huang*, 100 F.3d 135, 140 (Fed. Cir. 1996).

Thus, the Board employs a two-step analysis in evaluating whether a patentee has established a nexus between the evidence concerning objective indicia of nonobviousness and the merits of the claimed invention. See *Lectrosonics, Inc. v. Zaxcom, Inc.*, IPR2018-01129, Paper 33 at 32–33

(PTAB Jan. 24, 2020) (precedential). Initially, the Board considers whether the patentee has demonstrated that the “products are coextensive (or nearly coextensive) with the challenged claims,” resulting in a rebuttable presumption of nexus. *Id.* at 33. Absent a presumption of nexus, the Board considers whether the patentee has demonstrated “a legally and factually sufficient connection” between the evidence and the claimed invention. *See Henny Penny*, 938 F.3d at 1332; *Lectrosonics*, IPR2018-01129, Paper 33 at 33.

(ii) Presumption of Nexus

Patent Owner asserts that the BitRaider software embodies the claimed invention and “has been commercially successful.” Resp. 54; *see* Ex. 2105 § IV ¶¶ 22–28.

Petitioner asserts that Patent Owner “cites no evidence” that the BitRaider software “practices any claim” and, therefore, fails to show that the software is coextensive with any claim. Reply 25 (citing Resp. 53–56).

Based on the evidence of record, Patent Owner has not demonstrated that the BitRaider software is “coextensive (or nearly coextensive) with the challenged claims.” *See* Resp. 53–56; Ex. 2105 § IV ¶¶ 24, 26; *Lectrosonics*, IPR2018-01129, Paper 33 at 33. In his declaration, Mr. O’Brien states that the BitRaider software “incorporates the technology described and claimed in the ’808 patent.” Ex. 2105 § IV ¶ 24; *see id.* § IV ¶ 26. But Mr. O’Brien fails to map any ’808 patent claim to the BitRaider software and fails to explain how any software feature corresponds to any claim limitation. *See id.* § IV ¶¶ 24–28. Conclusory testimony “does not qualify as substantial evidence.” *TQ Delta, LLC v. Cisco Sys., Inc.*, 942 F.3d 1352, 1358 (Fed. Cir. 2019). Moreover, Patent Owner does not in the

Response map any '808 patent claim to the BitRaider software or explain how any software feature corresponds to any claim limitation. *See* Resp. 54–56. Nor has Patent Owner shown that the unclaimed features in the BitRaider software lack significance. *See id.* at 53–56; *infra* § III.D.3(p)(iii). Hence, Patent Owner has not established a rebuttable presumption of nexus.

(iii) Proof of Nexus: Commercial Success

Commercial success tends to show nonobviousness because “the law presumes an idea would successfully have been brought to market sooner, in response to market forces, had the idea been obvious to persons skilled in the art.” *Merck & Co. v. Teva Pharm. USA, Inc.*, 395 F.3d 1364, 1376 (Fed. Cir. 2005). Patent Owner asserts that the BitRaider software “has been commercially successful” and has resulted in licensing revenues exceeding \$2 million. Resp. 54; *see* Ex. 2105 § IV ¶¶ 22–28. Patent Owner identifies the “popular and well-received” game “Star Wars: The Old Republic” from Electronic Arts Inc. as a licensed product. Resp. 54; *see* Ex. 2105 § IV ¶¶ 25–26; Ex. 2130, 87–101 (Technology License Agreement Between BitRaider MMO, LLC and Electronic Arts Inc.). Patent Owner also asserts that the claimed invention permits “superior performance.” Resp. 51 (citing *In re Chupp*, 816 F.2d 643, 646 (Fed. Cir. 1987)).

Petitioner notes Patent Owner’s contentions that (1) “the video gaming industry exceeds \$10 billion per year” and (2) the BitRaider software “has earned \$2 million in licensing revenues, total, since 2007.” Reply 26 (citing Resp. 4, 54). Petitioner asserts that Patent Owner “at best” contends that licensed products “have earned about 0.0014 percent of the annual video gaming industry revenue—which is not ‘significant sales.’” *Id.*

Petitioner further asserts that Patent Owner fails to show that licensing revenues resulted from “any patented aspect” of the BitRaider software “rather than from some unpatented aspect of the software.” *Id.*

We agree with Petitioner that Patent Owner fails to show that significant sales or licensing revenues resulted from “any patented aspect” of the BitRaider software “rather than from some unpatented aspect of the software.” *See* Resp. 54–55; Reply 26. Among other things, Patent Owner fails to identify any license that expressly grants rights under the ’808 patent. Resp. 54–55; *see* Ex. 2105 § IV ¶¶ 22–28; Tr. 29:7–20.

In the licenses provided by Patent Owner, the licensed technology concerns source code or object code (or both) for many software features including the following: BRWC client executable; 32-bit filesystem minifilter; 64-bit filesystem minifilter; external pipe communications DLL; graphics overlay DLL; process injection executable; support services controller executable; BRProfiler base profiling tool; recognizer software for the game specific object log creation; RemotePacker authoring tool; PatchMaker patch creation tool; external pipe communication header files; external UI status monitoring header files; and runtime PDB debugging symbol files. *See, e.g.*, Ex. 2130, 1, 12, 19, 37–39, 67, 81, 87, 97, 112, 118, 128, 143, 179, 190, 197, 207, 215, 227, 233, 245. For instance, the licensed software features permit “progressive patching during gameplay” and “character movement prediction about twenty times per second.”

Ex. 2125, 2. The ’808 patent does not disclose or claim “character movement prediction about twenty times per second.” *See, e.g.*, Ex. 1001, 2:38–3:39, 12:44–14:54, code (57).

Patent Owner fails to explain which of the many licensed software features relate to the claimed invention. Resp. 54–55; *see* Ex. 2105 § IV ¶¶ 22–28. Thus, Patent Owner fails to show that any commercial success resulted directly from “the unique characteristics of the claimed invention” rather than other factors “unrelated to the quality of the patented subject matter.” *See* Resp. 54–55; *Huang*, 100 F.3d at 140. Patent Owner also fails to show that any license “arose out of recognition and acceptance of the subject matter claimed” in the ’808 patent. *See* Resp. 54–55; *In re GPAC Inc.*, 57 F.3d 1573, 1580 (Fed. Cir. 1995).

Additionally, Patent Owner cites no evidence supporting the assertion that the claimed invention permits “superior performance.” *See* Resp. 51. And Petitioner’s evidence of complaints about the BitRaider software refutes Patent Owner’s assertion about “superior performance.” *See* Ex. 1036, 1; Ex. 1037, 2.

(iv) Proof of Nexus: Industry Praise

Industry praise tends to show nonobviousness because “[i]ndustry participants, especially competitors, are not likely to praise an obvious advance over the known art.” *WBIP, LLC v. Kohler Co.*, 829 F.3d 1317, 1334 (Fed. Cir. 2016). Patent Owner asserts that the BitRaider software “has received predominant praise from professionals in the industry.” Resp. 55. Patent Owner also asserts that the BitRaider software “was acknowledged as a reason for success of” the game “Star Wars: The Old Republic.” *Id.* at 56. To support those assertions, Patent Owner cites the following: (1) a May 2009 news article in Engadget titled “Frogster Announces Plan to Stream Runes of Magic” (Exhibit 2124, “Frogster article”); (2) an October 2011 press release titled “GDC Online 2011: New

3D Virtual World VIE™: Virtual Island of Entertainment™ Integrates BitRaider Streaming Solution on BigWorld Technolog” (Exhibit 2125, “VIE press release”); and (3) a September 2013 press release titled “Star Wars™: The Old Republic™ Continues to Grow with New PVP Warzone Arenas Coming in Game Update 2.4: The Dread of War” (Exhibit 2126, “Star Wars press release”). *Id.* (citing Exs. 2124–2126).

Petitioner argues that the Frogster article, the VIE press release, and the Star Wars press release “provide no cognizable evidence of commercial success or praise” for the claimed invention. Reply 26. Specifically, Petitioner asserts that the Frogster article discusses “desired results from a contemplated future use” of the BitRaider software. *Id.* (emphases omitted). Petitioner asserts that the VIE press release comes from Mr. O’Brien and that “[s]elf-praise does not qualify as evidence of industry praise.” *Id.* Petitioner asserts that the Star Wars press release “attributes no sales growth to BitRaider and identifies several reasons that could have led to the game’s alleged revenue,” including the following:

- “many gameplay additions”;
- “the game’s first Digital Expansion: Rise of the Hutt Cartel for free”;
- “two new Operations and the launch of PvP Warzone Arenas”;
- “updates to the game every eight weeks by adding high-value social, solo and group content”;
- a new “epic finale”; and
- a “full live-action video.”

Id. at 26–27 (quoting Ex. 2126, 1).

Petitioner also argues that the BitRaider software “was a commercial disaster.” Reply 27. Specifically, Petitioner asserts that an Internet search for “BitRaider” results in “predominant complaints.” *Id.* As examples, Petitioner quotes statements that the BitRaider software (1) “drastically slow[s] down update speeds”; (2) “often causes ‘unspecified error’ among other numbered errors, and gets itself stuck ‘reorganizing data’ in a seemingly endless and futile loop”; and (3) “creates very slow speeds for many people.” *Id.* at 27–28 (alteration by Petitioner) (quoting Ex. 1036, 1; Ex. 1037, 2).

For the reasons stated by Petitioner, we agree with Petitioner that the Frogster article, the VIE press release, and the Star Wars press release provide little, if any, evidence of industry praise for the claimed invention. *See* Reply 26–27; Exs. 2124–2126. For instance, the Frogster article, the VIE press release, and the Star Wars press release reference the BitRaider software, not the claimed invention or the ’808 patent. Ex. 2124, 1; Ex. 2125, 1; Ex. 2126, 1. Patent Owner fails to explain which of the many BitRaider software features relate to the claimed invention. Resp. 55–56; *see* Ex. 2105 § IV ¶¶ 22–28; Ex. 2130, 12, 37–39, 81, 97, 112, 128, 143, 190, 207, 227, 245. Thus, Patent Owner fails to show that any industry praise resulted directly from “the unique characteristics of the claimed invention” rather than other factors “unrelated to the quality of the patented subject matter.” *See* Resp. 55–56; *Huang*, 100 F.3d at 140.

Additionally, Petitioner’s evidence of complaints about the BitRaider software refutes Patent Owner’s assertions about industry praise. *See* Resp. 55–56; Reply 27–28; Ex. 1036, 1; Ex. 1037, 2.

(v) Proof of Nexus: Long-Felt but Unmet Need

A long-felt but unmet need tends to show nonobviousness because “it is reasonable to infer that the need would not have persisted had the solution been obvious.” *WBIP*, 829 F.3d at 1332. In his declaration, Mr. O’Brien states that there was a long-felt need for “progressive downloading” of gaming applications. Ex. 2105 § IV ¶¶ 11, 23, 31. But evidence submitted by Patent Owner shows that others in the industry successfully provided software for “progressive downloading” of gaming applications. As an example, Exhibit 2118 states that software from Spoon (“a leader in cloud computing and virtualization technology”) delivered “desktop games on-demand with no installs via a small browser plugin” and ran “games using a hybrid of remote and local computing resources.” Ex. 2118, 1. As another example, Exhibit 2119 states that software from Numecent (a “[c]loud startup”) divided a game “into small fragments,” and “[w]hen that fragment is needed by the user,” the software sent “only that fragment on an on-demand basis” permitting the fragment to start running immediately “without the need for installation.” Ex. 2119, 3–4.

Also, a long-felt but unmet need “is analyzed as of the date of an articulated identified problem and evidence of efforts to solve that problem.” *Tex. Instrs. Inc. v. Int’l Trade Comm’n*, 988 F.2d 1165, 1178 (Fed. Cir. 1993). In his declaration, Mr. O’Brien does not discuss an “articulated identified problem” concerning “progressive downloading” of gaming applications using a minifilter as claimed. *See* Ex. 2105 § IV ¶¶ 7–20; *see also WBIP*, 829 F.3d at 1334 (discussing efforts to solve an “articulated identified problem” by an alleged infringer before the invention); *Apple Inc. v. Samsung Elecs. Co.*, 816 F.3d 788, 804–05 (Fed. Cir. 2016) (requiring

evidence of unsuccessful efforts to solve an “articulated identified problem” before the invention), *vacated in part on other grounds*, 839 F.3d 1034 (Fed. Cir. 2016); *Tex. Instrs.*, 988 F.2d at 1178 (discussing efforts to solve an “articulated identified problem” by others in the industry before the invention).

Moreover, Patent Owner does not in the Response rely on long-felt but unmet need as evidence of nonobviousness. *See* Resp. 53–56.

(vi) Proof of Nexus: Copying

Copying tends to show nonobviousness as “another form of flattering praise for inventive features.” *WBIP*, 829 F.3d at 1336. In his declaration, Mr. O’Brien states that Microsoft, Google, and Sony have developed products implementing “technology described and claimed in the ’808 patent” and “have copied the patented technology.” Ex. 2105 § IV ¶¶ 29–30. But Mr. O’Brien fails to map any ’808 patent claim to any Microsoft, Google, or Sony product. *Id.* § IV ¶¶ 29–31. “[E]vidence of copying . . . is legally irrelevant unless the [copied product] is shown to be an embodiment of the claims.” *Amazon.com, Inc. v. Barnesandnoble.com, Inc.*, 239 F.3d 1343, 1366 (Fed. Cir. 2001). Mr. O’Brien also fails to compare the BitRaider software to any Microsoft, Google, or Sony software. Ex. 2105 § IV ¶¶ 29–31. Copying “requires the replication of a specific product” or process. *Iron Grip Barbell Co. v. USA Sports, Inc.*, 392 F.3d 1317, 1325 (Fed. Cir. 2004); *see Liqwd, Inc. v. L’Oreal USA, Inc.*, 941 F.3d 1133, 1137–38 (Fed. Cir. 2019).

Moreover, Patent Owner does not in the Response rely on copying as evidence of nonobviousness. *See* Resp. 53–56.

(vii) Summary for Objective Indicia of Nonobviousness

For the reasons discussed above, Patent Owner has not shown that the BitRaider software “embodies the claimed features, and is coextensive with them.” *See* Resp. 53–56; Ex. 2105 § IV ¶¶ 24, 26; *supra* § III.D.3(p)(ii); *Fox Factory*, 944 F.3d at 1373. Nor has Patent Owner established a nexus between the evidence concerning objective indicia of nonobviousness and the merits of the claimed invention. *See* Resp. 53–56; Ex. 2105 § IV ¶¶ 22–31; *supra* § III.D.3(p)(iii)–(vi). Thus, we accord little weight to that evidence. But even if that evidence carried more weight, it would not outweigh the other evidence considered in the *Graham* analysis. *See Leapfrog Enters., Inc. v. Fisher-Price, Inc.*, 485 F.3d 1157, 1162 (Fed. Cir. 2007) (affirming an obviousness determination “given the strength of the prima facie obviousness showing” even though the patentee “provided substantial evidence of commercial success, praise, and long-felt need”).

(q) Conclusion About Obviousness/Nonobviousness for Claim 1

For the reasons discussed above, the combined disclosures in Christiansen and Eylon teach claim 1’s subject matter. *See supra* §§ III.D.3(a)–(n). Moreover, an ordinarily skilled artisan would have been motivated to combine Eylon’s teachings with Christiansen’s teachings as Petitioner proposes and would have had a reasonable expectation of success. *See supra* § III.D.3(o). Also, we accord little weight to Patent Owner’s evidence concerning objective indicia of nonobviousness. *See supra* § III.D.3(p). Hence, Petitioner has shown by a preponderance of the evidence that claim 1 is unpatentable under § 103(a) as obvious over Christiansen and Eylon.

4. INDEPENDENT CLAIM 14

As Patent Owner acknowledges, method claim 14 includes the same limitations as system claim 1 with one exception involving limitation [1.b.ii]. *See* Ex. 1001, 12:44–67, 13:54–14:24; Resp. 36. Claim 1 recites “said on-demand requester object being configured to receive I/O requests on behalf of an application for which data is available in data packs for streaming delivery.” Ex. 1001, 12:45–48. Claim 14 recites “said on-demand requester object being configured to receive I/O requests **from the I/O manager** on behalf of an application for which data is available in data packs for streaming delivery.” *Id.* at 14:1–4 (emphasis added). Thus, claim 14 includes a requirement that claim 1 lacks, i.e., that the on-demand requester object receive I/O requests “from the I/O manager.”

Petitioner contends that claim 14 is unpatentable under § 103(a) as obvious over Christiansen and Eylon for essentially the same reasons as claim 1. *See* Pet. 5, 30–48, 62–65. Regarding claim 14’s requirement that the on-demand requester object receive I/O requests “from the I/O manager,” Petitioner asserts that Christiansen “discloses that managed filters receive I/O requests from the ‘I/O Manager,’ in exactly the same way as” the ’808 patent. *Id.* at 63 (citing Ex. 1001, 5:59–66, Fig. 2; Ex. 1003 ¶ 391; Ex. 1006 ¶¶ 32–33, 35, 40, Figs. 2–4).

For claim 14, Patent Owner “incorporates the arguments made relative to” claim 1. Resp. 37. Further, Patent Owner asserts that “[w]ithout the VFS, there is no teaching to suggest a distinct sensor to determine when the requested data is present locally.” *Id.* Patent Owner also asserts that “without the means of distinguishing data which exists locally from remote

data, all that will issue from I/O manager are undifferentiated calls which the Filter Manager cannot treat in a differentiated manner.” *Id.*

Petitioner responds to Patent Owner’s assertions by explaining that the proposed combination includes Eylon’s VFS 160 and VFS 160 “is used to determine when the requested data is present locally.” Reply 17 (citing Pet. 47–48).

We agree with Petitioner that the combined disclosures in Christiansen and Eylon teach claim 14’s subject matter. *See* Pet. 30–48, 62–65; Ex. 1003 ¶¶ 148–256, 386–413; *supra* §§ III.D.3(a)–(n). Regarding claim 14’s requirement that the on-demand requester object receive I/O requests “from the I/O manager,” Christiansen discloses a computer system with an I/O stack including I/O manager 215 and filter manager 220 that service file system 225. Ex. 1006 ¶¶ 10, 32–33, Fig. 2; *see* Ex. 1003 ¶¶ 202–203. After receiving a file system request, I/O manager 215 “determine[s] what I/O request or requests should be issued to fulfill” the file system request and “send[s] each I/O request” to filter manager 220. Ex. 1006 ¶ 33. Filter manager 220 passes the callback data “associated with a particular type of I/O request to each registered filter sequentially in an order in which the registered filters are ordered.” *Id.* ¶ 37. Thus, in the proposed combination a managed filter receives I/O requests “from the I/O manager” via the filter manager according to claim 14. *See* Ex. 1003 ¶¶ 388–391.

As for Patent Owner’s assertions, the proposed combination implements Eylon’s streaming functionality as a managed filter in Christiansen’s system. *See* Pet. 24–25; Reply 20 & n.12; Ex. 1003 ¶¶ 127–128. Eylon’s streaming functionality includes VFS 160. Ex. 1005,

8:14–41, Fig. 4; *see id.* at 4:1–36; Ex. 1003 ¶¶ 120, 158. As Petitioner explains, VFS 160 “is used to determine when the requested data is present locally.” Ex. 1005, 4:17–28, 8:20–29; *see id.* at 10:43–46, 10:53–56, code (57), Fig. 6A; Ex. 1003 ¶¶ 123–124, 160, 168; *supra* § III.D.3(l); Reply 17.

Also, as discussed above for limitation [1.c.v], in the proposed combination a managed filter would have been “further configured to reference a table that includes names and paths with at least one offset address and length where each data pack required to fulfill each I/O request is located.” *See* Ex. 1003 ¶¶ 228–238; *supra* § III.D.3(l). In that table, the names and paths would have identified locally stored information and remotely stored information. *See* Ex. 1005, 11:42–51, Fig. 7; Ex. 1006 ¶¶ 44, 57–58; *supra* § III.D.3(l); *see also* Ex. 1003 ¶¶ 127, 142–145, 233–234.

For the reasons discussed above, the combined disclosures in Christiansen and Eylon teach claim 14’s subject matter. *See supra* §§ III.D.3(a)–(n). Moreover, an ordinarily skilled artisan would have been motivated to combine Eylon’s teachings with Christiansen’s teachings as Petitioner proposes and would have had a reasonable expectation of success. *See supra* § III.D.3(o). Also, we accord little weight to Patent Owner’s evidence concerning objective indicia of nonobviousness. *See supra* § III.D.3(p). Hence, Petitioner has shown by a preponderance of the evidence that claim 14 is unpatentable under § 103(a) as obvious over Christiansen and Eylon.

5. DEPENDENT CLAIMS 2–13 AND 15–20

(a) Dependent Claim 2

Claim 2 depends directly from claim 1 and recites “said minifilter being configured to load before compression and filter minifilter drivers in the I/O stack.” Ex. 1001, 13:1–3. As discussed above, we construe “filter minifilter driver” to include “an encryption minifilter driver.” See Inst. Dec. 28; *supra* § III.C.4.

Petitioner contends that the combined disclosures in Christiansen and Eylon teach claim 2’s subject matter. See Pet. 48–49. Specifically, Petitioner asserts that Eylon’s streaming support system 102 functions to “locate and return requested streamlets.” *Id.* at 48. Petitioner asserts that with Eylon’s streaming functionality implemented as a managed filter in Christiansen’s system a requested streamlet “must” by “logical necessity” be “located before the streamlet can be compressed/decompressed or encrypted/decrypted.” *Id.* (citing Ex. 1003 ¶ 260; Ex. 1007, 9:7–14; Ex. 1026 ¶ 31). Petitioner also asserts that “assigning the streaming support system minifilter a higher altitude than compression and encryption minifilters would simplify the design of the minifilter” for two related reasons. See *id.* at 48–49. In particular, at a higher altitude than compression and encryption minifilters, the streaming support system minifilter (1) “would receive I/O requests unmodified by the compression and encryption minifilters” and, therefore, (2) “would not need to address the complexity of handling modified requests from compression and encryption minifilters.” *Id.* at 48–49 (citing Ex. 1003 ¶ 261). Petitioner further asserts that these reasons would have motivated an ordinarily skilled artisan to load Eylon’s streaming support system 102 “at a

higher altitude than compression and encryption minifilters” in the combined system Petitioner proposes. *Id.* at 49 (citing Ex. 1003 ¶¶ 262–263).

Patent Owner contends that claim 2 is not obvious over the references because claim 1 is not obvious over the references. Resp. 38.

For the reasons stated by Petitioner and supported by Dr. Houh’s testimony as well as Mr. Catlin’s testimony, we agree with Petitioner that the combined disclosures in Christiansen and Eylon teach claim 2’s subject matter. *See* Pet. 48–49; Ex. 1003 ¶¶ 257–263; Ex. 1026 ¶¶ 30–33.

(b) Dependent Claims 3 and 15

Claim 3 depends directly from claim 1 and recites “said minifilter attaching to the I/O stack by registering with a filter manager in the I/O stack for I/O requests for the application for which data is available in data packs for streaming delivery.” Ex. 1001, 13:4–8. Claim 15 depends directly from claim 14 and recites subject matter similar to claim 3. *Id.* at 14:25–29.

Petitioner contends that the combined disclosures in Christiansen and Eylon teach the subject matter of claims 3 and 15. *See* Pet. 32–33, 39–42, 49, 65. Specifically, Petitioner asserts that claims 3 and 15 “rehash” limitations [1.b.ii], [1.c.i], and [1.c.iii] and further require that “registering with a filter manager is the way a minifilter ‘attach[es] to the I/O stack.’” *Id.* at 49 (alteration by Petitioner) (emphasis omitted). Petitioner also asserts that Christiansen “discloses that the minifilter attaches to the I/O stack by registering with the filter manager for I/O requests in which the filter is interested in receiving.” *Id.* (citing Ex. 1003 ¶ 287; Ex. 1006 ¶¶ 34, 38, 44).

Patent Owner contends that claims 3 and 15 are not obvious over the references because claims 1 and 14 are not obvious over the references. Resp. 38.

For the reasons stated by Petitioner and supported by Dr. Houh’s testimony, we agree with Petitioner that the combined disclosures in Christiansen and Eylon teach the subject matter of claims 3 and 15. *See* Pet. 32–33, 39–42, 49, 65; Ex. 1003 ¶¶ 160–165, 198–207, 213–220, 280–287, 414–415; Ex. 1006 ¶¶ 10, 32–34, 38, 44, Fig. 2; Ex. 1035 ¶¶ 19, 22; *supra* §§ III.D.3(c), III.D.3(h), III.D.3(j). For example, Christiansen discloses a computer system with an I/O stack including I/O manager 215 and filter manager 220 that service file system 225. Ex. 1006 ¶¶ 10, 32–33, Fig. 2; *see* Ex. 1003 ¶¶ 202–203. Christiansen also discloses that a managed filter attaches to the I/O stack by registering with filter manager 220 “to be called and receive callback data for I/Os related to” a storage volume in file system 225. Ex. 1006 ¶¶ 38, 44; *see id.* ¶ 34; Ex. 1003 ¶¶ 284–286.

(c) Dependent Claim 4

Claim 4 depends directly from claim 1 and recites “said table including a size indicator for each data pack.” Ex. 1001, 13:9–10.

Petitioner contends that Eylon teaches a “table including a size indicator for each data pack” because the differences between the offset addresses in Eylon’s streamlet map 220 determine a “length” for each streamlet corresponding to a “size indicator” for each “data pack.” Pet. 44, 49. Petitioner also contends that the ’808 patent equates a “length” to a “size.” *Id.* at 49 (citing Ex. 1001, 7:29–32).

Patent Owner makes no arguments specific to claim 4. *See, e.g.*, Resp. 23–48; Sur-reply 13–17. Nonetheless, the burden remains on Petitioner to demonstrate unpatentability. *See Dynamic Drinkware*, 800 F.3d at 1378.

We agree with Petitioner that Eylon teaches a “table including a size indicator for each data pack” because the differences between the offset addresses in Eylon’s streamlet map 220 determine a “length” for each streamlet corresponding to a “size indicator” for each “data pack.” *See* Pet. 44, 49; Ex. 1003 ¶¶ 230–231, 288–292; Ex. 1005, 12:4–16, Figs. 7–8; *supra* § III.D.3(1). Consistent with this, the ’808 patent explains that “start and end offsets” may serve as a “size indicator.” Ex. 1001, 6:44–47; *see id.* at 7:29–32. Thus, the combined disclosures in Christiansen and Eylon teach claim 4’s subject matter. *See* Ex. 1003 ¶¶ 288–292.

(d) Dependent Claims 5 and 16

Claim 5 depends directly from claim 1 and recites “said minifilter being configured to pass on a read call if a required data pack is available locally and if all data in the required data pack has been received.” Ex. 1001, 13:11–14. Claim 16 depends directly from claim 14 and recites subject matter similar to claim 5. *Id.* at 14:30–33.

Petitioner contends that the combined disclosures in Christiansen and Eylon teach the subject matter of claims 5 and 16. *See* Pet. 49–52, 65. Specifically, Petitioner asserts that Christiansen teaches a managed filter “configured to pass on a read call” if the required information is “available locally” for two related reasons. *Id.* at 49–50. First, Christiansen discloses a managed filter with a mapping between locally stored information and remotely stored information. *Id.* at 49. Second, Christiansen discloses directing an I/O request for locally stored information to a local file system at the bottom of the I/O stack. *Id.* at 49–50 (citing Ex. 1006 ¶¶ 57, 68–69). According to Petitioner, directing an I/O request for locally stored information to a local file system corresponds to “pass[ing] on a read call”

if the required information is “available locally.” *Id.* at 50 (citing Ex. 1003 ¶ 296).

Additionally, Petitioner asserts that Eylon teaches determining in response to a read call (1) “if a required data pack is available locally” and (2) “if all data in the required data pack has been received.” Pet. 50–51. In particular, Petitioner contends that Eylon discloses determining in response to a read call if “the requested streamlets are present” in VFS 160, i.e., “available locally” on the client. *Id.* at 50 (citing Ex. 1003 ¶ 299; Ex. 1005, 4:10–25). Petitioner also contends that Eylon discloses verifying streamlet integrity, i.e., that a streamlet “has been received properly and not corrupted.” *Id.* at 51 (citing Ex. 1005, 11:20–28). According to Petitioner, verification includes “ensuring that all portions of the streamlet had been properly received, otherwise the integrity of the streamlet could not be conclusively determined.” *Id.* (citing Ex. 1003 ¶ 301).

Patent Owner disputes that the combined disclosures in Christiansen and Eylon teach the subject matter of claims 5 and 16. *See* Resp. 39. Specifically, Patent Owner asserts that in the proposed combination “there is no supplied way to detect the presence of the data locally.” *Id.* Patent Owner also asserts that Christiansen discloses a minifilter residing only “on the server and not on the client.” *Id.*

For the reasons stated by Petitioner and supported by Dr. Houh’s testimony, we agree with Petitioner that the combined disclosures in Christiansen and Eylon teach the subject matter of claims 5 and 16. *See* Pet. 49–52, 65; Ex. 1003 ¶¶ 127, 142–145, 233–234, 293–303, 416–417; Ex. 1005, 4:10–25, 8:35–37, 9:50–57, 11:20–28, 11:40–51, 12:32–42, Fig. 5; Ex. 1006 ¶¶ 44, 57–58, 68–69, code (57).

We disagree with Patent Owner’s assertion that in the proposed combination “there is no supplied way to detect the presence of the data locally.” *See* Resp. 39. As discussed above, the proposed combination includes a table with names and paths identifying locally stored information and remotely stored information. *See* Ex. 1005, 11:42–51, Fig. 7; Ex. 1006 ¶¶ 44, 57–58; *see also* Ex. 1003 ¶¶ 127, 142–145, 233–234; *supra* §§ III.D.3(l), III.D.4.

We also disagree with Patent Owner’s assertion that Christiansen discloses a minifilter residing only “on the server and not on the client.” *See* Resp. 39. As discussed above, Christiansen discloses that a filter may reside on a server or a client (or both). *See* Ex. 1006 ¶¶ 21, 39, 43; Ex. 1035 ¶ 22; *supra* § III.D.3(h).

(e) Dependent Claims 6 and 17

Claim 6 depends directly from claim 1 and recites “said minifilter being configured to withhold a read call if a required data pack is not available locally.” Ex. 1001, 13:15–17. Claim 17 depends directly from claim 14 and recites subject matter similar to claim 6. *Id.* at 14:34–36.

Petitioner contends that Christiansen and Eylon each teach “withhold a read call” if required data “is not available locally.” *See* Pet. 52–53, 65. First, Petitioner asserts that Christiansen discloses that “where the requested information is not available locally, the I/O request ‘is sent to the remote file system,’” thus withholding the I/O request from the local file system. *Id.* at 52 (quoting Ex. 1006 ¶ 72; citing Ex. 1003 ¶ 309).

Second, Petitioner asserts that Eylon discloses that “[i]f a needed streamlet is not present, a streamlet request is issued to the server and the virtual file system maintains a busy status until the necessary streamlets have

been provided.” Pet. 52 (alteration by Petitioner) (emphasis omitted) (quoting Ex. 1005, code (57)). According to Petitioner, maintaining a “busy status until the necessary streamlets have been provided” means that the file system driver and the virtual file system “withhold the read call because the read call cannot be passed on and fulfilled by the operating system until the data is available in local memory.” *Id.* (citing Ex. 1003 ¶ 309).

Patent Owner disputes that the combined disclosures in Christiansen and Eylon teach the subject matter of claims 6 and 17. *See* Resp. 39–40. Specifically, Patent Owner asserts that each reference lacks “a trigger to determine whether to withhold a read call.” *Id.* at 39.

For the reasons stated by Petitioner and supported by Dr. Houh’s testimony, we agree with Petitioner that Christiansen and Eylon each teach “withhold a read call” if required data “is not available locally.” *See* Pet. 52; Ex. 1003 ¶¶ 305–310; Ex. 1005, 4:23–30, code (57); Ex. 1006 ¶¶ 44, 72, Fig. 9. Thus, the combined disclosures in Christiansen and Eylon teach the subject matter of claims 6 and 17. *See* Ex. 1003 ¶¶ 305–310, 418–419.

We disagree with Patent Owner’s assertion that each reference lacks “a trigger to determine whether to withhold a read call” because each reference includes such a trigger. *See* Resp. 39. First, Christiansen discloses determining whether an I/O request seeks locally stored information or remotely stored information. Ex. 1006 ¶¶ 54, 57–58, 66–72, Fig. 9; *see* Ex. 1003 ¶¶ 116, 309. If the requested information “is not available locally,” the I/O request “is sent to the remote file system.” Ex. 1006 ¶ 72; *see* Ex. 1003 ¶ 309. Hence, contrary to Patent Owner’s assertion, the requested information’s unavailability in the local file system triggers withholding a read call and sending the I/O request to the remote file system.

Second, as discussed above, Eylon discloses determining if “the requested streamlets are present” in VFS 160 when a streaming application executes. Ex. 1005, 4:23–25; *see id.* at 4:10–22, 9:50–61, 10:34–56, 15:22–27, 16:59–65, 17:47–53, code (57), Fig. 6A; *supra* § III.D.3(n). If “one or more of the requested streamlet blocks are absent” from VFS 160, “a fetch request is issued to the server for the appropriate streamlet blocks.” Ex. 1005, 4:25–28; *see id.* at 10:53–56, 11:1–10, 13:2–4, 15:28–32, 17:1–7, 17:56–59, code (57), Fig. 6A; Ex. 1003 ¶¶ 124, 168; Ex. 1035 ¶ 20; *supra* § III.D.3(n). Hence, contrary to Patent Owner’s assertion, the requested information’s unavailability locally in VFS 160 triggers withholding a read call and issuing a fetch request for remotely stored information.

(f) Dependent Claims 7 and 18

Claim 7 depends directly from claim 1 and recites “said minifilter being configured to withhold a read call if all data in the required data pack has not been received.” Ex. 1001, 13:18–20. Claim 18 depends directly from claim 14 and recites subject matter similar to claim 7. *Id.* at 14:37–39.

Petitioner contends that Eylon teaches “withhold a read call if all data in the required data pack has not been received.” *See* Pet. 53–54, 65. Specifically, as discussed above for claims 5 and 16, Petitioner asserts that Eylon teaches determining in response to a read call “if all data in the required data pack has been received.” *See id.* at 51, 53; *supra* § III.D.5(d). Petitioner also asserts that Eylon discloses maintaining a “busy status” if a requested streamlet is “not present” and that an ordinarily skilled artisan would have understood that disclosure “to mean that one or more of that streamlet’s constituent data is missing.” Pet. 53 (quoting Ex. 1005, code (57); citing Ex. 1003 ¶ 316).

Patent Owner disputes that the combined disclosures in Christiansen and Eylon teach the subject matter of claims 7 and 18. *See* Resp. 40. Specifically, Patent Owner asserts that “the only reason” for configuring a minifilter according to claims 7 and 18 “is to tread the path of the inventor, constructing the combination in the confirmed view of hindsight.” *Id.*

For the reasons stated by Petitioner and supported by Dr. Houh’s testimony, we agree with Petitioner that Eylon teaches “withhold a read call if all data in the required data pack has not been received.” *See* Pet. 51, 53, 65; Ex. 1003 ¶¶ 301–302, 312–316, 420–421; Ex. 1005, 4:10–28, 11:20–28, 12:32–37, 17:1–7, code (57); *supra* § III.D.5(d). Thus, the combined disclosures in Christiansen and Eylon teach the subject matter of claims 7 and 18. *See* Ex. 1003 ¶¶ 312–316, 420–421.

We disagree with Patent Owner’s assertion that “the only reason” for configuring a minifilter according to claims 7 and 18 “is to tread the path of the inventor, constructing the combination in the confirmed view of hindsight.” *See* Resp. 40. Eylon discloses that “when a streamlet is received at the client,” the streamlet is processed, e.g., by initially verifying “the data integrity of the streamlet.” Ex. 1005, 11:20–24, Fig. 6B; *see id.* at 12:32–37; Ex. 1003 ¶¶ 246–247, 301–302. Verification includes “ensuring that all portions of the streamlet had been properly received.” Ex. 1003 ¶ 301. If a streamlet fails verification, e.g., due to incomplete or corrupt data, streaming support system 102 discards the streamlet, and VFS 160 lacks the streamlet data. Ex. 1005, 11:24–28, Fig. 6B; *see id.* at 12:32–37; Ex. 1003 ¶¶ 246–247. And if VFS 160 lacks the streamlet data for a requested streamlet, VFS 160 maintains a “busy status,” and “a fetch request is issued to the server.” Ex. 1005, 4:25–28, 17:1–7, code (57); *see id.* at 10:53–56;

Ex. 1003 ¶¶ 306–307, 314. VFS 160 maintains a “busy status” until the server provides the necessary data. Ex. 1005, code (57); *see id.* at 17:1–7; Ex. 1003 ¶¶ 306, 314. By maintaining a “busy status,” VFS 160 “withhold[s] a read call if all data in the required data pack has not been received” according to claims 7 and 18. Ex. 1003 ¶¶ 308, 315.

Also, as discussed above, Dr. Houh provides a detailed explanation why an ordinarily skilled artisan would have had sound reasons to combine Eylon’s teachings with Christiansen’s teachings as Petitioner proposes. *See* Ex. 1003 ¶¶ 129–146; *supra* § III.D.3(o). That detailed explanation demonstrates that Dr. Houh did not fall prey to hindsight bias. *See supra* § III.D.3(o).

(g) Dependent Claim 8

Claim 8 depends directly from claim 1 and recites “said at least one address including an[] address for at least one local source for a first data pack and an address for at least one remote source for a second data pack.” Ex. 1001, 13:21–24.

Petitioner contends that the combined disclosures in Christiansen and Eylon teach claim 8’s subject matter. *See* Pet. 54–57. Specifically, Petitioner asserts that Christiansen discloses a mapping table with addresses for locally stored files and remotely stored files. *Id.* at 54–55 (citing Ex. 1006 ¶¶ 54–55, 57–58). Petitioner asserts that Eylon discloses a streaming application using files comprising locally stored streamlets and remotely stored streamlets. *Id.* at 54–55. Petitioner explains that in the proposed combination the mapping table’s addresses identify files for Eylon’s streaming application. *Id.* at 54. Petitioner asserts that each file constitutes a “source” for the streamlets comprising the file. *Id.* at 54–55.

To support Petitioner’s assertions, Dr. Houh testifies that when a file or part of a file “is stored locally, the local address for this file is an address for the source of each of its constituent streamlets, including the first one (‘a first data pack’).” Ex. 1003 ¶ 326 (emphasis omitted); *see* Pet. 55. He also testifies that when a file or part of a file “is stored remotely, the remote address for this file is an address for the source of each of its constituent streamlets, including the second one (‘a second data pack’).” Ex. 1003 ¶ 326 (emphasis omitted); *see* Pet. 55.

Patent Owner disputes that the combined disclosures in Christiansen and Eylon teach claim 8’s subject matter. *See* Resp. 40–41. Specifically, Patent Owner asserts that Christiansen discloses a minifilter residing only “on the server” and, therefore, “remote is the client machine.” *Id.* at 40–41 (citing Ex. 1006 ¶¶ 54–55, 57–58).

For the reasons stated by Petitioner and supported by Dr. Houh’s testimony, we agree with Petitioner that the combined disclosures in Christiansen and Eylon teach claim 8’s subject matter. *See* Pet. 54–55; Ex. 1003 ¶¶ 319–327; Ex. 1005, 12:1–16, Figs. 7–8; Ex. 1006 ¶¶ 54–55, 57–58.

We disagree with Patent Owner’s assertion that Christiansen discloses a minifilter residing only “on the server” and, therefore, “remote is the client machine.” *See* Resp. 40–41. As discussed above, Christiansen discloses that a filter may reside on a server or a client (or both). *See* Ex. 1006 ¶¶ 21, 39, 43; Ex. 1035 ¶ 22; *supra* § III.D.3(h).

(h) Dependent Claims 9, 10, and 19

Claims 9, 10, and 19 read as follows (with formatting added for clarity):

9. A streaming on demand system according to claim 1, said at least one address including

an[] address for at least one local source for a first data pack, said local source being a source from the group consisting of a hard disk, an optical storage medium, a nonvolatile memory, a magnetic storage medium, and

an address for at least one remote source for a second data pack.

10. A streaming on demand system according to claim 1, said at least one address including

an[] address for at least one local source for a first data pack, and

an address for at least one remote source for a second data pack, said remote source being a source from the group consisting of a server and a peer computing device.

19. A streaming on demand method according to claim 14, said at least one address including

an[] address for at least one local source for a first data pack, said local source being a source from the group consisting of a hard disk, an optical storage medium, a nonvolatile memory, a magnetic storage medium, and

an address for at least one remote source for a second data pack, said remote source being a source from the group consisting of a server and a peer computing device.

Ex. 1001, 13:25–37, 14:40–48.

Petitioner contends that the combined disclosures in Christiansen and Eylon teach the subject matter of claims 9, 10, and 19 for the same reasons the combined disclosures teach claim 8’s subject matter and because Christiansen discloses a “local source” from the recited local “group” and a “remote source” from the recited remote “group.” *See* Pet. 57–58, 65 (citing Ex. 1006 ¶¶ 23–24, 26, 28); *see id.* at 54–55.

Patent Owner contends that claims 9, 10, and 19 are not obvious over the references because claims 1 and 14 are not obvious over the references. Resp. 41.

We agree with Petitioner that the combined disclosures in Christiansen and Eylon teach the subject matter of claims 9, 10, and 19. *See* Pet. 57–58, 65; Ex. 1003 ¶¶ 333–342, 422–423. For the reasons discussed above, the combined disclosures in Christiansen and Eylon teach claim 8’s subject matter, including “an[] address for at least one local source for a first data pack” and “an address for at least one remote source for a second data pack” as recited in each of claims 9, 10, and 19. *See* Ex. 1003 ¶¶ 319–327; Ex. 1005, 12:1–16, Figs. 7–8; Ex. 1006 ¶¶ 54–55, 57–58; *supra* § III.D.5(g).

Additionally, Christiansen teaches a “local source” from “the group consisting of a hard disk, an optical storage medium, a nonvolatile memory, [and] a magnetic storage medium” as recited in each of claims 9 and 19 because Christiansen discloses a local computer including, among other things, “a hard disk drive,” “an optical disk drive,” and “a magnetic disk drive.” Ex. 1006 ¶ 26, Fig. 1; *see id.* ¶¶ 23–24, 28; Ex. 1003 ¶¶ 335–337. Christiansen also teaches a “remote source” from “the group consisting of a server and a peer computing device” as recited in each of claims 10 and 19 because Christiansen discloses a local computer connected via a network to a remote computer, such as “a server” or “a peer device.” Ex. 1006 ¶ 28; *see* Ex. 1003 ¶¶ 340–342.

(i) Dependent Claims 11 and 20

Claim 11 depends directly from claim 1 and recites “said minifilter being configured to request to retrieve a required data pack from a first source corresponding to a first address in the table if the data pack is

available from the first source, and to identify an alternative address in the table if the data pack is not available from the first source.” Ex. 1001, 13:38–44. Claim 20 depends directly from claim 14 and recites subject matter similar to claim 11. *Id.* at 14:49–54.

Petitioner contends that the combined disclosures in Christiansen and Eylon teach the subject matter of claims 11 and 20. *See* Pet. 58–60, 65. Specifically, Petitioner asserts that Christiansen teaches a “minifilter being configured to request to retrieve a required data pack from a first source corresponding to a first address in the table if the data pack is available from the first source” for two related reasons. *Id.* at 58–59. First, Christiansen discloses determining in response to an I/O request whether a local cache contains the requested information, e.g., by “obtaining the cache directory from the mapping table” and attempting to open a file. *Id.* at 58 (quoting Ex. 1006 ¶ 68). Second, Christiansen discloses sending the I/O request “to the local file system” if the local cache contains the requested information. *Id.* at 58–59 (quoting Ex. 1006 ¶ 69). Petitioner explains that these operations correspond to a “request to retrieve a required data pack from a first source” and that, for the reasons discussed for claim 8, locally cached information “is available from the first source” and will have “a first address” according to claims 11 and 20. *Id.* at 58–59; *see id.* at 55.

Petitioner asserts that Eylon teaches identifying “an alternative address” if “the data pack is not available from the first source” according to claims 11 and 20 because Eylon discloses that streaming support system 102 “issues a fetch request” to a remote server for a missing streamlet. Pet. 59 (citing Ex. 1005, 4:25–30, code (57)). Petitioner explains that the combined system would have looked up “in the mapping table” the missing streamlet’s

“address on the remote server.” *Id.* at 59–60 (citing Ex. 1003 ¶¶ 350–353; Ex. 1005, 10:53–56; Ex. 1006 ¶¶ 57–58).

Patent Owner contends that claims 11 and 20 are not obvious over the references because claims 1 and 14 are not obvious over the references. Resp. 41.

For the reasons stated by Petitioner and supported by Dr. Houh’s testimony, we agree with Petitioner that the combined disclosures in Christiansen and Eylon teach the subject matter of claims 11 and 20. *See* Pet. 55, 58–60; Ex. 1003 ¶¶ 324–326, 343–353, 424–425; Ex. 1005, 4:25–30, 10:53–56, 13:2–4, 15:28–32, 17:1–7, 17:56–59, code (57), Fig. 6A; Ex. 1006 ¶¶ 57–74, Figs. 8–9.

(j) Dependent Claim 12

Claim 12 depends directly from claim 1 and recites “said minifilter being configured, if the I/O request is a write operation, to mark an area as delivered and then write an address and size indicator corresponding to the write operation to the table.” Ex. 1001, 13:45–49.

Petitioner contends that the combined disclosures in Christiansen and Eylon teach claim 12’s subject matter. *See* Pet. 60–61. Specifically, Petitioner asserts that with Eylon’s streaming functionality implemented as a managed filter in Christiansen’s system the combined system would have had the ability “to mark an area as delivered” because Eylon discloses a map indicating “which streamlets are present in and which are absent from” local storage. *Id.* at 60 (citing Ex. 1005, 12:1–29, Fig. 7).

Petitioner asserts that it “would have been obvious to use this functionality in the combined system in conjunction with a write operation, and to additionally ‘write an address and size indicator corresponding to the

write operation to the table’ in response to that operation” for two related reasons. Pet. 60–61. First, Christiansen discloses that a write operation (1) produces modifications to remotely stored data but not locally stored data and (2) causes a remote system to notify the “applications that are interested in the target data” so the applications can delete their locally stored copies of the target data. *Id.* at 60 (citing Ex. 1003 ¶ 356; Ex. 1006 ¶¶ 61–62, 75–78, Fig. 10). Second, an ordinarily skilled artisan would have been motivated to configure the combined system to “immediately read” the modified data back into local storage and mark the modified data “as present” to “speed up access” and save time “when the application next seeks” the modified data. *Id.* at 61 (citing Ex. 1003 ¶ 358; Ex. 1005, 12:1–30, Fig. 7). According to Petitioner, an ordinarily skilled artisan would have known that “applications are more likely to access recently accessed data.” *Id.* (citing Ex. 1029, 473–74; Ex. 1031, 62–64).

For claim 12, Patent Owner “adopts his prior assertions relative to Eylon and Christiansen as asserted relative to” claim 1. Resp. 42.

For the reasons stated by Petitioner and supported by Dr. Houh’s testimony, we agree with Petitioner that the combined disclosures in Christiansen and Eylon teach claim 12’s subject matter. *See* Pet. 60–61; Ex. 1003 ¶¶ 354–358; Ex. 1005, 12:1–30, Fig. 7; Ex. 1006 ¶¶ 61–62, 75–78, Fig. 10; Ex. 1029, 473–74; Ex. 1031, 62–64.

(k) Dependent Claim 13

Claim 13 depends directly from claim 1 and recites “said minifilter being configured, if the I/O request is a create operation, to attach extended data to a file stream, said extended data corresponding to the create operation.” Ex. 1001, 13:50–53.

Petitioner contends that the combined disclosures in Christiansen and Eylon teach claim 13's subject matter. *See* Pet. 60–61; Reply 20–21; Paper 28, 1–3. Specifically, Petitioner asserts that Christiansen discloses determining in response to an I/O request whether a local cache contains a requested object. Paper 28, 1 (citing Ex. 1006 ¶¶ 59–60, Fig. 8). Petitioner further asserts that “[i]f the requested object is cached locally, Christiansen’s minifilter provides a ‘new name’ for the local version of the object to an I/O manager, which name was obtained from a reparse point associated with that object.” *Id.* at 1–2 (citing Ex. 1006 ¶¶ 68, 80–81, Fig. 9).

Additionally, Petitioner contends that the new name is “extended data” and that the locally cached object is a “file stream.” Paper 28, 2. Petitioner also contends that in the proposed combination a locally cached Eylon streamlet is a “file stream.” *Id.* According to Petitioner, the new name “is attached/associated with” the locally cached object because the new name “is used by the file system as the name for” the locally cached object “in responding to that I/O request create operation.” *Id.* at 3 (citing Ex. 1006 ¶ 68).

For claim 13, Patent Owner “adopts his prior assertion relative to Eylon and Christiansen as asserted against” claim 1. Resp. 45–46.

We disagree with Petitioner that the combined disclosures in Christiansen and Eylon teach claim 13's subject matter. *See* Pet. 60–61; Reply 20–21; Paper 28, 1–3. Christiansen discloses determining whether a local cache contains a requested object by using a mapping table containing remote file-system names mapped to local cache-directory names for locally cached objects. Ex. 1006 ¶¶ 57–58, 67–68, Fig. 9. Christiansen explains that for a create operation that seeks to “open an already-existing object,” the

“I/O is reparsed to the new name” meaning that the “I/O is redirected to the cached object of the local file system.” *Id.* ¶ 68; *see id.* ¶¶ 60, 65, Fig. 8 (step 835), Fig. 9. That create operation does not cause a minifilter to “attach extended data to a file stream” as required by claim 13 because that create operation does not cause the new name to become associated with the locally cached object. The new name was previously associated with the locally cached object when generating the mapping table containing remote file-system names mapped to local cache-directory names for locally cached objects. *See* Ex. 1006 ¶¶ 57–58, 67–68, Fig. 9.

We also disagree with Petitioner’s contention that a locally cached Eylon streamlet is a “file stream.” *See* Paper 28, 2. A locally cached Eylon streamlet constitutes file data, not a “data structure for file data” as we have construed “file stream.” *See* Ex. 1005, 3:57–59, 5:53–54, code (57); *supra* § III.C.6.

(l) Conclusion About Obviousness/Nonobviousness

For the reasons discussed above, the combined disclosures in Christiansen and Eylon teach the subject matter of claims 2–12 and 15–20. *See supra* §§ III.D.5(a)–(j). Moreover, an ordinarily skilled artisan would have been motivated to combine Eylon’s teachings with Christiansen’s teachings as Petitioner proposes and would have had a reasonable expectation of success. *See supra* § III.D.3(o). Also, we accord little weight to Patent Owner’s evidence concerning objective indicia of nonobviousness. *See supra* § III.D.3(p). Additionally, Patent Owner does not argue that any objective indicia of nonobviousness resulted directly from an allegedly unique characteristic of a dependent claim. *See* Resp. 53–56. Hence, Petitioner has shown by a preponderance of the evidence that claims 2–12

and 15–20 are unpatentable under § 103(a) as obvious over Christiansen and Eylon. But Petitioner has not shown by a preponderance of the evidence that claim 13 is unpatentable under § 103(a) as obvious over Christiansen and Eylon. *See supra* § III.D.5(k).

*E. Alleged Obviousness over Christiansen,
Eylon, and Pudipeddi: Claim 2*

Petitioner contends that claim 2 is unpatentable under § 103(a) as obvious over Christiansen, Eylon, and Pudipeddi. *See* Pet. 5, 65–68. Above, we provided overviews of Christiansen and Eylon. *See supra* §§ III.D.1–III.D.2. Below, we provide an overview of Pudipeddi, and then we consider the obviousness issues raised by the parties. For the reasons explained below, we agree with Petitioner that claim 2 is unpatentable under § 103(a) as obvious over Christiansen, Eylon, and Pudipeddi.

1. OVERVIEW OF PUDIPEDDI (EXHIBIT 1007)

Pudipeddi is a U.S. patent titled “Managed File System Filter Model and Architecture,” filed on December 9, 2002, and issued on January 31, 2006. Ex. 1007, codes (22), (45), (54). Pudipeddi explains that file-system filters “enhance the underlying file system by performing various file-related computing tasks that users desire, including tasks such as passing file system I/O (requests and data) through anti-virus software, file system quota providers, file replicators and encryption/compression products.” *Id.* at 1:18–23. Pudipeddi also explains that by installing filters “computer users can select the file system features they want and need, in a manner that enables upgrades, replacement, insertion, removal of the components without necessitating the changing the actual operating system or file system driver code.” *Id.* at 1:32–37.

But Pudipeddi identifies “drawbacks of the existing model” for file-system filters, such as redundancy and inefficiency. *See* Ex. 1007, 1:55–2:57. For instance, “an antivirus product can slow down a system as much as sixty percent, but not every I/O request received by an antivirus filter is one that the filter will do anything with, namely inspect any corresponding data for viruses.” *Id.* at 2:2–6.

To address those drawbacks, Pudipeddi “provides a model/ architecture” with filters “managed by a filter manager to receive callbacks for I/O requests” that the filters have registered with the filter manager. Ex. 1007, 2:31–34, code (57). Filters register only for particular I/O operations, such as one or more “create, read, write, [and] close” operations. *Id.* at 2:54–58, code (57). Further, filters “register two kinds of callbacks,” namely, pre-callbacks and post-callbacks. *Id.* at 9:20–22, 9:42–45, 9:59–67, code (57). A “pre-callback is called on the I/O’s way down, that is, towards the file system.” *Id.* at 9:22–23, code (57). A “post-callback is called during the completion of the I/O, on the way back up from the file system towards the I/O manager.” *Id.* at 9:24–26, code (57).

A filter manager associates a filter with an “altitude” indicating the filter’s place in the callback order. Ex. 1007, 2:65–3:8, 8:51–52, 9:7–11. For instance, an antivirus filter should not “attach between an encryption filter and the base file system” because the antivirus filter “needs to see the data as is, prior to encryption.” *Id.* at 9:11–14, 16:6–10. More generally, a filter examining data should not attach below a filter modifying data. *Id.* at 16:10–13, 16:19–38; *see id.* at 15:53–63.

To avoid problems due to improper placement in the callback order, Pudipeddi provides a table with a “logical ordering” for managed filters by

altitude. Ex. 1007, 16:19–38. In that “logical ordering,” (1) an antivirus filter has an altitude above a hierarchical-storage-management (HSM) filter, (2) an HSM filter has an altitude above a compression filter, (3) a compression filter has an altitude above an encryption filter, and (4) an encryption filter has an altitude above the base file system. *Id.* at 16:25–38.

Pudipeddi’s table with a “logical ordering” for managed filters is reproduced below (Ex. 1007, 16:19–38):

Activity Monitor (file spy etc.)
Undelete
Anti-virus
Replication
Continuous backup
Content screener
Quota management
Cluster file system
HSM (3 rd party hierarchical storage management)
Compression
Encryption
Physical Quota Management
Open File backup (snapshots of open files)
Security Enhancer
Copy protection
System
Filter Infrastructure (filter manager)

The table above lists various managed filters in altitude order from high at the table’s top to low at the table’s bottom.

2. DIFFERENCES BETWEEN THE CLAIMED SUBJECT MATTER AND THE PRIOR ART

Claim 2 depends directly from claim 1 and recites “said minifilter being configured to load before compression and filter minifilter drivers in the I/O stack.” Ex. 1001, 13:1–3. As discussed above, we construe “filter minifilter driver” to include “an encryption minifilter driver.” *See* Inst. Dec. 28; *supra* § III.C.4.

Petitioner contends that Pudipeddi teaches claim 2's limitations because Pudipeddi provides a table with a "logical ordering" for managed filters by altitude where an HSM filter has a higher altitude than compression and encryption filters. Pet. 66 (citing Ex. 1007, 16:21–38). To support Petitioner's contention, Dr. Houh testifies that an ordinarily skilled artisan would have interpreted Pudipeddi's table with a "logical ordering" for managed filters "as showing the ordering of filters by altitude from the top of the I/O stack to the bottom." Ex. 1003 ¶¶ 267–268. He also testifies that an ordinarily skilled artisan would have known that hierarchical storage management constitutes "a technique for managing the location and retrieval of files from remote locations, such as servers." *Id.* ¶ 268 (citing Ex. 1025 (article about hierarchical storage management); Ex. 1028, 1:13–24 (U.S. Patent No. 7,441,096 B2 to Kitamura)). He further testifies that Pudipeddi's table with a "logical ordering" for managed filters shows an HSM filter with "an altitude above the altitude of" compression and encryption filters. *Id.* ¶ 269.

Patent Owner contends that claim 2 is not obvious over the references because claim 1 is not obvious over the references. Resp. 38.

For the reasons stated by Petitioner and supported by Dr. Houh's testimony, we agree with Petitioner that Pudipeddi teaches claim 2's limitations. *See* Pet. 66; Ex. 1003 ¶¶ 266–269. Specifically, Pudipeddi provides a table with a "logical ordering" for managed filters by altitude. Ex. 1007, 16:19–38; *see* Ex. 1003 ¶ 267. In that "logical ordering," an HSM filter has an altitude above a compression filter, and a compression filter has an altitude above an encryption filter. Ex. 1007, 16:25–32; *see* Ex. 1003 ¶¶ 268–269. Hence, the HSM filter loads before the compression and

encryption filters in the I/O stack as required by claim 2. *See* Ex. 1003 ¶ 269; *supra* § III.C.4.

3. ALLEGED REASONS FOR COMBINING THE TEACHINGS OF THE REFERENCES

Petitioner contends that an ordinarily skilled artisan would have been motivated to combine Pudipeddi’s filter ordering with the Christiansen-Eylon system for several related reasons. *See* Pet. 67–68. First, Petitioner asserts that Pudipeddi discloses a “logical ordering” for managed filters by altitude. *Id.* at 68 (citing Ex. 1007, 16:19–20). Second, Petitioner asserts that an ordinarily skilled artisan would have recognized the “logical necessity” of ordering a data-locating filter above “filters for compressing/decompressing or encrypting/decrypting” data. *Id.* (citing Ex. 1003 ¶ 260; Ex. 1007, 9:7–14; Ex. 1026 ¶ 31); *see id.* at 48. Third, Petitioner asserts that such an ordering would have simplified filter design because the data-locating filter “would receive I/O requests unmodified by the compression and encryption minifilters, and would not need to address the complexity of handling modified requests from compression and encryption minifilters.” *Id.* at 48–49, 68 (citing Ex. 1003 ¶¶ 261–262).

To support Petitioner’s position, Dr. Houh testifies that an ordinarily skilled artisan would have considered Eylon’s streaming support system 102 analogous to Pudipeddi’s HSM filter because streaming support system 102 “manages the location of and retrieving files from remote locations such as servers.” Ex. 1003 ¶ 273. He also testifies that an ordinarily skilled artisan would have understood that ordering streaming support system 102 “at the HSM level” above compression and encryption filters “is advantageous” for two reasons. *Id.* ¶ 278; *see id.* ¶¶ 260–261. First, a different ordering

“would risk causing the compression and encryption minifilters to not function” because “the requested streamlet must be located before the streamlet can be compressed/decompressed or encrypted/decrypted.” *Id.* ¶¶ 260, 278 (citing Ex. 1007, 9:7–14). Second, “the exemplary ordering simplifies the design of” streaming support system 102 “as it does not need to handle I/O requests modified by the compression and encryption minifilters.” *Id.* ¶ 278; *see id.* ¶ 261.

Petitioner also contends that the proposed combination “would have been only the arrangement of old elements” and would have yielded “no more” than an ordinarily skilled artisan “would expect from such an arrangement.” Pet. 67 (citing Ex. 1003 ¶ 272). To support Petitioner’s contention, Dr. Houh testifies that an ordinarily skilled artisan would have had a reasonable expectation of success combining Pudipeddi’s filter ordering with the Christiansen-Eylon system because the proposed combination “would have been only the arrangement of old elements” and would have yielded “no more” than an ordinarily skilled artisan “would expect from such an arrangement.” Ex. 1003 ¶ 272.

Patent Owner makes no arguments against combining Pudipeddi’s teachings with the teachings of Christiansen and Eylon. *See, e.g.*, Resp. 37–38; Sur-reply 9–17. Nonetheless, the burden remains on Petitioner to demonstrate unpatentability. *See Dynamic Drinkware*, 800 F.3d at 1378.

For the reasons stated by Petitioner and supported by Dr. Houh’s testimony, we agree with Petitioner that an ordinarily skilled artisan would have been motivated to combine Pudipeddi’s filter ordering with the Christiansen-Eylon system. *See* Pet. 48–49, 67–68; Ex. 1003 ¶¶ 260–261, 273–279; Ex. 1007, 9:11–14, 16:6–10, 16:19–38; *see also* Ex. 1026

¶¶ 30–33. We also agree that an ordinarily skilled artisan would have had a reasonable expectation of success implementing the proposed combination. *See* Pet. 67; Ex. 1003 ¶ 272.

4. CONCLUSION ABOUT OBVIOUSNESS/NONOBVIOUSNESS FOR CLAIM 2

For the reasons discussed above, the combined disclosures in Christiansen, Eylon, and Pudipeddi teach claim 2’s subject matter. *See supra* §§ III.D.3(a)–(n), III.E.2. Moreover, an ordinarily skilled artisan would have been motivated to combine Pudipeddi’s teachings with the teachings of Christiansen and Eylon as Petitioner proposes and would have had a reasonable expectation of success. *See supra* § III.E.3. Also, we accord little weight to Patent Owner’s evidence concerning objective indicia of nonobviousness. *See supra* § III.D.3(p). Additionally, Patent Owner does not argue that any objective indicia of nonobviousness resulted directly from an allegedly unique characteristic of claim 2. *See* Resp. 53–56. Hence, Petitioner has shown by a preponderance of the evidence that claim 2 is unpatentable under § 103(a) as obvious over Christiansen, Eylon, and Pudipeddi.

F. Alleged Obviousness over Christiansen, Eylon, and Prahlad: Claim 12

Petitioner contends that claim 12 is unpatentable under § 103(a) as obvious over Christiansen, Eylon, and Prahlad. *See* Pet. 5, 68–71. Above, we provided overviews of Christiansen and Eylon. *See supra* §§ III.D.1–III.D.2. Below, we provide an overview of Prahlad, and then we consider the obviousness issues raised by the parties. For the reasons explained below, we agree with Petitioner that claim 12 is unpatentable under § 103(a) as obvious over Christiansen, Eylon, and Prahlad.

1. OVERVIEW OF PRAHLAD (EXHIBIT 1016)

Prahlad is a U.S. patent titled “Destination Systems and Methods for Performing Data Replication,” filed on December 18, 2006, and issued on November 10, 2009.⁶ Ex. 1016, codes (22), (45), (54). Prahlad states that the invention relates to “data replication in a storage management system.” *Id.* at 1:43–46, code (57).

Prahlad explains that “the success or failure of an important transaction may turn on the availability of information that is both accurate and current” and that computer users “seek reliable, cost-effective ways to protect the information stored on their computer networks.” Ex. 1016, 1:53–58. Prahlad also explains that many “approaches to protecting data involve creating a copy of the data, such as backing up and/or replicating data on one or more storage devices.” *Id.* at 1:59–61. Prahlad describes deficiencies in various approaches involving copying data. *Id.* at 1:67–2:36. Those deficiencies include (1) expending processing resources “in copying data to a destination system rather than being used to process application requests” and (2) requiring “a tremendous amount of storage space.” *Id.* at 2:13–15, 2:28–36. Prahlad identifies a need to “reduce the impact (e.g., processing load) on a source, or primary, system when performing one or more data management and/or storage operations on data, such as, for example, application-specific data.” *Id.* at 2:42–46.

⁶ In the Prior Publication Data, Prahlad identifies US 2007/0185937 A1, published on August 9, 2007. Ex. 1016, code (65).

Prahlad's Figure 1 (reproduced below) illustrates a replication system that endeavors to address that need:

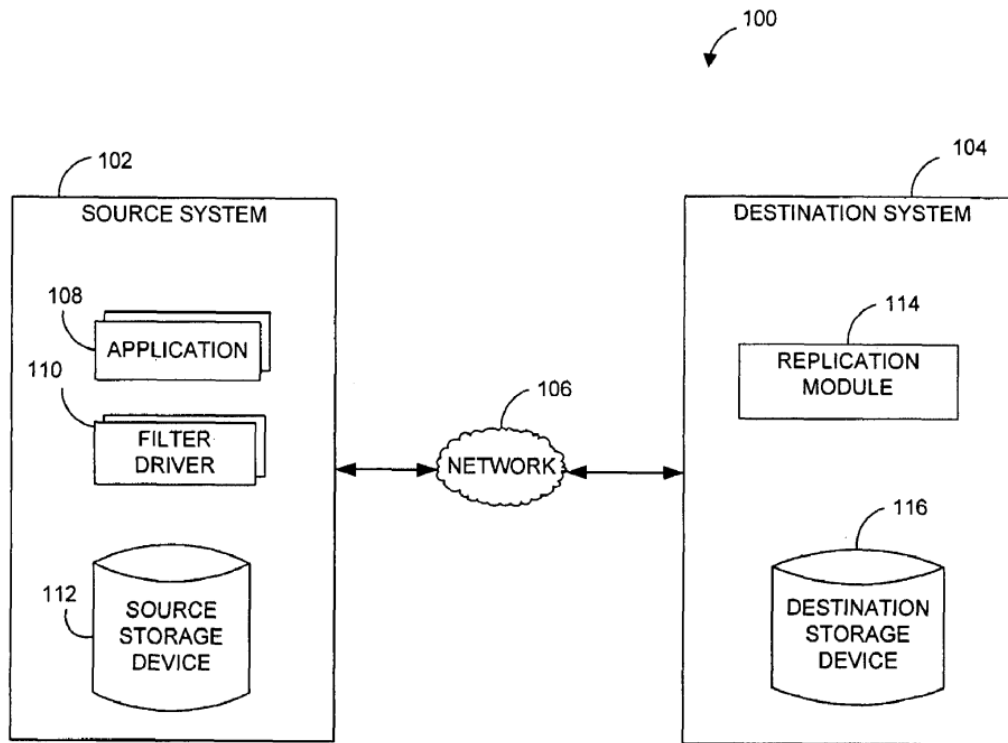


FIGURE 1

Figure 1, above, is a block diagram depicting replication system 100 with source system 102 and destination system 104 connected via network 106. Ex. 1016, 10:50–58, Fig. 1; *see id.* at 9:8–9.

Source system 102 includes one or more applications 108 that reside on a computing device, one or more filter drivers 110 that “interact with data (e.g., production data) associated with” applications 108, and source storage device 112. Ex. 1016, 11:8–23, 11:65–67, Fig. 1; *see id.* at 15:21–25, Fig. 2A. Destination system 104 includes replication module 114 and destination storage device 116. *Id.* at 12:7–33, Fig. 1; *see id.* at 17:62–18:3, Fig. 3.

Among other things, a filter driver may “monitor[] and/or intercept[] particular application requests targeted at a file system.” Ex. 1016, 11:24–36; *see id.* at 15:26–31, 15:45–16:36, Fig. 2A. For example, a filter driver may intercept data-modification operations that “include changes, updates and new information (e.g., data writes) with respect to the application(s) 108 of interest.” *Id.* at 11:37–40; *see id.* at 16:8–17. After intercepting a data-modification operation, a filter driver may populate a log to indicate the changes made by the operation. *Id.* at 15:56–59; *see id.* at 16:30–34, 19:4–18, 20:42–56.

Prahlad’s Figure 5 (reproduced below) illustrates a data structure for a log entry usable in Figure 1’s replication system:

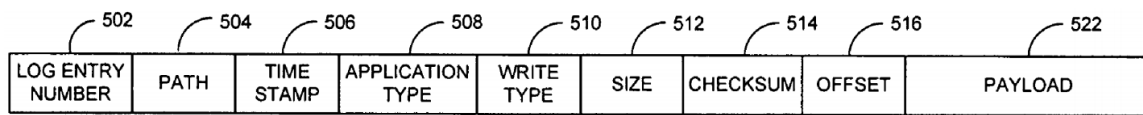


FIGURE 5

Figure 5, above, is a block diagram depicting various fields in a data structure for a log entry usable in Figure 1’s replication system. Ex. 1016, 9:21–23, 27:12–14, Fig. 5. In particular, Figure 5’s log entry relates to “a data write operation in the replication system.” *Id.* at 27:20–22; *see id.* at 27:62–63, 28:24–26, 28:37–39.

More generally, a log entry “comprises information regarding modifications to data and/or files on the source storage device 112 and may include, for example, information regarding: which file was modified, the time of the modification, the type of modification, the relative data, a unique identification, combinations of the same or the like.” Ex. 1016, 27:14–20;

see id. at 28:55–29:12. A filter driver may generate a log entry. *Id.* at 27:23–24, 27:66–28:1.

Figure 5’s log entry includes the following fields: “a log entry number field 502, a path field 504, a time stamp field 506, an application type field 508, a write type field 510, a size field 512, a checksum field 514, an offset field 516 and a payload field 522.” Ex. 1016, 27:26–30; *see id.* at 27:31–28:54. “The size field 512 may include information relating to the size (e.g., the number of bytes) of the data being modified by the data write operation.” *Id.* at 28:24–26. “The offset field 516 may include information relating to the location within a file or portion of data that the data write is occurring.” *Id.* at 28:37–39.

2. DIFFERENCES BETWEEN THE CLAIMED SUBJECT MATTER AND THE PRIOR ART

Claim 12 depends directly from claim 1 and recites “said minifilter being configured, if the I/O request is a write operation, to mark an area as delivered and then write an address and size indicator corresponding to the write operation to the table.” Ex. 1001, 13:45–49.

Petitioner contends that Prahlad teaches claim 12’s limitations because Prahlad discloses generating a log entry for a write operation that includes an offset field and a size field. *See* Pet. 69–70 (citing Ex. 1003 ¶ 365; Ex. 1016, 27:26–30, 27:62–63, 28:24–26, 28:37–39, Fig. 5). Petitioner also contends that generating a log entry according to Prahlad “constitutes ‘mark[ing] an area as delivered’ because it indicates that modified data for the target of the write operation has been delivered to the storage device.” *Id.* at 70 (alteration by Petitioner) (emphasis omitted) (quoting Ex. 1001, 13:47 (claim 12)).

Patent Owner disputes that Prahlad teaches claim 12's limitations. *See* Resp. 43, 45. Specifically, Patent Owner asserts that Prahlad "fails to disclose" limitation [1.b.i], i.e., "an on-demand requester object installed on a computing device." *Id.* at 43; *see id.* at 45. Patent Owner asserts that Prahlad "simply time stamps data write operations of a computer application and generates log entries to reflect a last known good state." *Id.* at 43. Patent Owner also asserts that Petitioner "provides no means for moving from logged entries to the [memory lookup audit table] MLAT with its offset field and a size field." *Id.*

For the reasons stated by Petitioner and supported by Dr. Houh's testimony, we agree with Petitioner that Prahlad teaches claim 12's limitations. *See* Pet. 69–70; Ex. 1003 ¶¶ 363–365. Specifically, Prahlad discloses generating a log entry for a write operation. Ex. 1016, 27:20–22. As Petitioner contends, generating a log entry according to Prahlad constitutes "mark[ing] an area as delivered" according to claim 12 because "it indicates that modified data for the target of the write operation has been delivered to the storage device." *See* Pet. 70; Ex. 1003 ¶ 365.

Furthermore, as Prahlad's Figure 5 illustrates, a log entry includes an offset field with location information and a size field with size information. Ex. 1016, 27:26–30, 28:24–26, 28:37–39, Fig. 5. The '808 patent indicates that an offset may serve as an address. *See* Ex. 1001, 7:29–32, 7:38–42, 7:54–56, 7:61–65, 8:32–47, 9:28–29, Fig. 4. Thus, Prahlad teaches "writ[ing] an address and size indicator corresponding to the write operation to [a] table" according to claim 12. *See* Ex. 1003 ¶¶ 363–365.

Patent Owner's assertion that Prahlad "fails to disclose" limitation [1.b.i], i.e., "an on-demand requester object installed on a computing

device,” does not distinguish claim 12 from the references because Petitioner relies on Eylon, not Prahlad, for teaching limitation [1.b.i]. *See* Pet. 31–32; Resp. 43, 45; Reply 19; *supra* § III.D.3(b).

Patent Owner’s assertion that Petitioner “provides no means for moving from logged entries to the [memory lookup audit table] MLAT with its offset field and a size field” does not distinguish claim 12 from the references because Petitioner relies on Eylon, not Prahlad, for teaching the claimed “table” with “at least one offset address and length” for each data pack. *See* Pet. 43–46; Resp. 43; *supra* § III.D.3(l). Prahlad’s disclosure that a log entry includes an offset field with location (address) information and a size field with size (length) information demonstrates Prahlad’s compatibility with the Christiansen-Eylon system. *See* Ex. 1016, 27:26–30, 28:24–26, 28:37–39, Fig. 5; *see also* Ex. 1003 ¶¶ 363–364, 369.

3. ALLEGED REASONS FOR COMBINING THE TEACHINGS OF THE REFERENCES

Petitioner contends that an ordinarily skilled artisan would have been motivated to combine Prahlad’s logging functionality for write operations with Eylon’s streaming functionality because Prahlad’s logging functionality would have provided “an efficient way of ensuring that updated data written to a file in response to a write operation would not be overwritten with outdated data in response to subsequent I/O requests.” Pet. 70–71 (citing Ex. 1003 ¶ 371; Ex. 1016, 1:48–67, 2:40–46, 9:59–63).

To support Petitioner’s contention, Dr. Houh testifies that:

- (1) Eylon’s file system driver (FSD) 150 and virtual file system (VFS) 160 “are primarily used to locate streamlets needed by the application”; and

- (2) Prahlad’s “filter driver is primarily used to track data modification operations to ensure data remains accurate and current.”

Ex. 1003 ¶ 371 (citing Ex. 1005, 4:10–28, code (57); Ex. 1016, 1:48–67, 2:40–46, 9:59–63). He also testifies that an ordinarily skilled artisan would have recognized that Prahlad’s logging functionality for write operations would have provided “an efficient way of ensuring that updated data written to a file in response to a write operation would not be overwritten with outdated data in response to subsequent I/O requests.” *Id.* Dr. Houh adds that Prahlad’s disclosure that the logging operations help “ensure that the replicated data remains ‘both accurate and current’ with minimal burden to the computer system” would have provided further motivation to include Prahlad’s logging functionality for write operations in the Christiansen-Eylon system. *Id.* ¶ 372 (quoting Ex. 1016, 1:54–55).

Petitioner also contends that the proposed combination “would have been only the arrangement of old elements” and would have yielded “no more” than an ordinarily skilled artisan “would expect from such an arrangement.” Pet. 70 (citing Ex. 1003 ¶ 369). To support Petitioner’s contention, Dr. Houh testifies that an ordinarily skilled artisan would have had a reasonable expectation of success combining Prahlad’s logging functionality for write operations with Eylon’s streaming functionality because the proposed combination “would have been only the arrangement of old elements” and would have yielded “no more” than an ordinarily skilled artisan “would expect from such an arrangement.” Ex. 1003 ¶ 369.

Patent Owner disputes that an ordinarily skilled artisan would have been motivated to combine Prahlad’s teachings with the teachings of Christiansen and Eylon. *See* Resp. 44–45. In particular, Patent Owner

contends that “there is no motivation to construct the combination with the functionality” Petitioner proposes because “Eylon gives the complete solution” and “the motivation for combining evaporates.” *Id.* at 44–45.

Petitioner responds to Patent Owner’s contention concerning “no motivation to construct the combination with the functionality” Petitioner proposes by asserting that Patent Owner “never explains how Eylon discloses the logging functionality for which Prahlad was cited.” Reply 20.

For the reasons stated by Petitioner and supported by Dr. Houh’s testimony, we agree with Petitioner that an ordinarily skilled artisan would have been motivated to combine Prahlad’s logging functionality for write operations with Eylon’s streaming functionality. *See* Pet. 70–71; Ex. 1003 ¶¶ 366–373; 1005, 4:10–28, code (57); Ex. 1016, 1:48–67, 2:40–46, 9:59–63. In particular, Prahlad’s logging functionality for write operations would have provided “an efficient way of ensuring that updated data written to a file in response to a write operation would not be overwritten with outdated data in response to subsequent I/O requests.” *See* Pet. 71; Ex. 1003 ¶¶ 370–371. We also agree that an ordinarily skilled artisan would have had a reasonable expectation of success implementing the proposed combination. *See* Pet. 70; Ex. 1003 ¶ 369.

As for Patent Owner’s contention concerning “no motivation to construct the combination with the functionality” Petitioner proposes, we agree with Petitioner that Patent Owner “never explains how Eylon discloses the logging functionality for which Prahlad was cited.” *See* Resp. 44–45; Reply 20; Sur-reply 1–17.

4. CONCLUSION ABOUT OBVIOUSNESS/NONOBVIOUSNESS FOR CLAIM 12

For the reasons discussed above, the combined disclosures in Christiansen, Eylon, and Prahlad teach claim 12's subject matter. *See supra* §§ III.D.3(a)–(n), III.F.2. Moreover, an ordinarily skilled artisan would have been motivated to combine Prahlad's teachings with Eylon's teachings as Petitioner proposes and would have had a reasonable expectation of success. *See supra* § III.F.3. Also, we accord little weight to Patent Owner's evidence concerning objective indicia of nonobviousness. *See supra* § III.D.3(p). Additionally, Patent Owner does not argue that any objective indicia of nonobviousness resulted directly from an allegedly unique characteristic of claim 12. *See Resp.* 53–56. Hence, Petitioner has shown by a preponderance of the evidence that claim 12 is unpatentable under § 103(a) as obvious over Christiansen, Eylon, and Prahlad.

G. Alleged Obviousness over Christiansen, Eylon, and Thind: Claim 13

Petitioner contends that claim 13 is unpatentable under § 103(a) as obvious over Christiansen, Eylon, and Thind. *See Pet.* 5, 72–74. Above, we provided overviews of Christiansen and Eylon. *See supra* §§ III.D.1–III.D.2. Below, we provide an overview of Thind, and then we consider the obviousness issues raised by the parties. For the reasons explained below, we agree with Petitioner that claim 13 is unpatentable under § 103(a) as obvious over Christiansen, Eylon, and Thind.

1. OVERVIEW OF THIND (EXHIBIT 1008)

Thind is a U.S. patent titled “Method and System for Maintaining Namespace Consistency with a File System,” filed on November 30, 2004,

and issued on February 24, 2009.⁷ Ex. 1008, codes (22), (45), (54). Both Thind and Christiansen name two common inventors, i.e., Ravinder S. Thind and Neal R. Christiansen. Ex. 1006, code (75); Ex. 1008, code (75). In addition, Thind and Christiansen contain identical drawings as Figures 2–4 as well as identical descriptions of those drawings. *Compare* Ex. 1006 ¶¶ 32–42, Figs. 2–4, *with* Ex. 1008, 5:49–7:22, Figs. 2–4.

Thind explains that a file-system filter “may maintain internal metadata for files and directories on a volume.” Ex. 1008, 1:39–40. Thind also explains that changes to a volume associated with a file-system filter “may cause the internal metadata of the filter to be out of sync with the state of the volume” and “may cause the filter to behave incorrectly or render it unable to perform its desired function.” *Id.* at 1:40–44.

Hence, Thind endeavors to preserve “namespace consistency between selected objects maintained by a file system and a filter associated therewith.” Ex. 1008, 1:45–47, 1:51–54, code (57). The objects may include “directories, files, [and] anything that may be stored on a file system.” *Id.* at 8:3–5; *see id.* at 9:18–20, 16:12–15.

Specifically, a “filter monitors selected types of requests (or operations associated therewith) and determines whether the object is within a namespace associated with the filter.” Ex. 1008, 1:55–57, code (57). Based on “a change to the object,” the “namespace associated with the filter is updated.” *Id.* at 1:57–59, code (57). For instance, the filter “may monitor deletes” and then “update its metadata” to (1) “avoid monitoring objects that have been deleted,” (2) “not enforce policies when an existing object is

⁷ In the Prior Publication Data, Thind identifies US 2006/0116985 A1, published on June 1, 2006. Ex. 1008, code (65).

deleted and a new object with the same name is recreated in its place,” and (3) “reduce the size of the metadata used in monitoring such objects.” *Id.* at 7:62–67.

Like Christiansen, Thind explains that a create operation “may create an object or open an already-existing object.” Ex. 1008, 12:13–14; *see* Ex. 1006 ¶ 60. Additionally, Thind discloses several types of delete-related operations, including a create-for-delete-on-close operation. Ex. 1008, 2:60–63, 3:1–8, 9:53–59, 12:4–13:19, 13:46–14:48, Figs. 6, 11–12, 14–15. For the create-for-delete-on-close operation, Thind explains that an object “may be created for delete on close,” and “[w]hen an object is created for delete on close, this means that the object should be deleted after the object is closed.” *Id.* at 12:14–17. Thus, an object “may be created, exist, and continue to exist, until all processes that have handles to the object have closed their handles.” *Id.* at 12:17–19. Then, “[a]fter all the handles have been closed, the object is deleted.” *Id.* at 12:19–20.

Thind discloses that the create-for-delete-on-close operation allocates a stream context to an object. Ex. 1008, 13:4–5, Fig. 12 (step 1225); *see id.* at 9:42–44. Thind explains that a stream context comprises “data into which information about an object may be stored and retrieved.” *Id.* at 9:40–42. For instance, “[t]o track the name of the object through renames, the new name of the object is stored in the stream context.” *Id.* at 9:45–47; *see id.* at 9:47–52.

2. DIFFERENCES BETWEEN THE CLAIMED SUBJECT MATTER AND THE PRIOR ART

Claim 13 depends directly from claim 1 and recites “said minifilter being configured, if the I/O request is a create operation, to attach extended

data to a file stream, said extended data corresponding to the create operation.” Ex. 1001, 13:50–53.

Petitioner contends that Thind teaches claim 13’s limitations because Thind discloses a create operation that (1) allocates a stream context to an object and (2) stores an object name and a delete flag in the stream context. Pet. 72 (citing Ex. 1008, 13:4–12, Fig. 12); Paper 28, 3, 5 (citing Ex. 1008, 13:4–12, Fig. 12). Petitioner also contends that Thind’s create operation attaches “extended data” to a “file stream” by storing an object name and a delete flag in the stream context allocated by the create operation for a file. Pet. 72; Paper 28, 5 & n.2. According to Petitioner, the object name and the delete flag are “extended data,” and the stream context is a “file stream.” Pet. 72; Paper 28, 4 & n.2.

Patent Owner disputes that Thind teaches claim 13’s limitations. *See* Resp. 46–48; Paper 27, 1–5; Paper 30, 2. Specifically, Patent Owner asserts that Thind “does not create extended data but creates a distinct file structure” called a “stream context” where a delete flag is stored. Resp. 47 (citing Ex. 1008, 9:39–52). Patent Owner asserts that Thind “describes the purpose of the stream context, i.e. ‘[t]o track the name of the object through renames, the new name of the object is stored in the stream context.’” Paper 27, 2 (quoting Ex. 1008, 9:45–47). Patent Owner also asserts that a stream context is (1) “a transient structure in dynamically allocated memory” and (2) “only meaningful where actual file names and directories are assigned” and “the file in question is being used.” *Id.* at 2–3 (citing Ex. 1010, 13); *see* Paper 30, 2.

Additionally, Patent Owner seeks to distinguish claim 13 from Thind by contending that Thind’s minifilter “displays no ability to retrieve ‘data

packs’ and assemble those ‘data packs’ into a ‘file stream.’” Paper 27, 4; *see* Paper 30, 2.

We agree with Petitioner that Thind teaches claim 13’s limitations. *See* Pet. 72; Reply 22–23; Paper 28, 3–5, 4 n.2; Ex. 1003 ¶ 379; Ex. 1035 ¶ 34. As Petitioner contends, Thind discloses a create operation that (1) allocates a stream context to an object and (2) stores an object name and a delete flag in the stream context. Ex. 1008, 9:40–49, 13:4–11, Fig. 12 (steps 1225, 1230, and 1235); *see id.* at 9:49–52; Ex. 1003 ¶ 379; Ex. 1035 ¶ 34; Pet. 72; Paper 28, 3.

Thind explains that a stream context comprises “data into which information about an object may be stored and retrieved,” i.e., a data structure for an object. Ex. 1008, 9:40–42; *see* Resp. 47 (quoting Ex. 1008, 9:39–52). Thind also explains that an object includes a file. Ex. 1008, 8:3–5, 16:14–15; *see* Ex. 1006 ¶ 46; Ex. 1035 ¶ 34; Paper 27, 4–5 (stating that Thind “allows that the object could be a file”). Consistent with these explanations, Patent Owner describes a stream context as a “distinct file structure.” Resp. 47. Also consistent with these explanations, the Robert reference (cited by the Examiner during prosecution) describes a stream context as a “structure that represents the physical file.” Ex. 1019 ¶ 75.

Hence, when Thind’s object is a file, a stream context for the file is a “data structure for file data,” and thus a “file stream” as we have construed that term. *See supra* § III.C.6. And when Thind’s object is a file, an object name and a delete flag for the file are “data related to a file,” and thus “extended data” as we have construed that term. *See supra* § III.C.5.

Thind’s create operation attaches “extended data” to a “file stream” by storing an object name and a delete flag in the stream context allocated by

the create operation for a file. Ex. 1008, 13:4–7, Fig. 12 (steps 1225, 1230, and 1235); *see id.* at 9:40–44; Ex. 1003 ¶ 379; Ex. 1035 ¶ 34. The object name and the delete flag “correspond[] to the create operation” because the create operation generates the name and the flag. *See* Ex. 1008, 13:5–7, Fig. 12 (steps 1230 and 1235).

We disagree with Patent Owner’s assertion that Thind “does not create extended data.” *See* Resp. 47. Thind discloses storing an object name and a delete flag in a stream context for a file, and the object name and the delete flag are “extended data.” Ex. 1008, 8:3–5, 9:40–49, 13:4–11, Fig. 12 (steps 1225, 1230, and 1235); Ex. 1003 ¶ 379; Ex. 1035 ¶ 34.

As for Patent Owner’s assertion that a stream context is (1) “a transient structure in dynamically allocated memory” and (2) “only meaningful where actual file names and directories are assigned” and “the file in question is being used,” claim 13 does not specify a duration for a “file stream.” *See* Paper 27, 2–3; Ex. 1001, 13:50–53. So “a transient structure in dynamically allocated memory” may serve as a “file stream.”

As for Patent Owner’s contention that Thind’s minifilter “displays no ability to retrieve ‘data packs’ and assemble those ‘data packs’ into a ‘file stream,’” claim 13 does not require that a minifilter “retrieve ‘data packs’ and assemble those ‘data packs’ into a ‘file stream.’” *See* Paper 27, 4; Ex. 1001, 13:50–53.

3. ALLEGED REASONS FOR COMBINING THE TEACHINGS OF THE REFERENCES

Petitioner contends that an ordinarily skilled artisan would have been motivated to combine Thind’s extended-data functionality with Christiansen’s create operation because Thind’s extended-data functionality

would have provided “several advantages.” *See* Pet. 72–73. To support Petitioner’s contention, Dr. Houh testifies that the following advantages of Thind’s extended-data functionality would have motivated an ordinarily skilled artisan to combine that functionality with Christiansen’s create operation: (1) “preventing the filter metadata from being out of sync” and (2) “maintaining the ability of the filter to behave properly and perform its intended function.” Ex. 1003 ¶ 384 (citing Ex. 1008, 1:40–44).

Additionally, Dr. Houh testifies that an ordinarily skilled artisan would have recognized that Thind “identifies and solves a problem that will arise” in Eylon’s system. Ex. 1003 ¶ 384. He explains that Eylon’s system “specifically contemplates” delete operations. *Id.* (citing Ex. 1005, 13:50–53). He also explains that Thind discloses that “its invention may be useful to prevent filter malfunction due to file system requests including delete operations.” *Id.* (citing Ex. 1008, 7:41–67).

Petitioner also contends that the proposed combination “would have been only the arrangement of old elements” and would have yielded “no more” than an ordinarily skilled artisan “would expect from such an arrangement.” Pet. 73 (citing Ex. 1003 ¶ 383). To support Petitioner’s contention, Dr. Houh testifies that an ordinarily skilled artisan would have had a reasonable expectation of success combining Thind’s extended-data functionality with Christiansen’s create operation because the proposed combination “would have been only the arrangement of old elements” and would have yielded “no more” than an ordinarily skilled artisan “would expect from such an arrangement.” Ex. 1003 ¶ 383.

Patent Owner disputes that an ordinarily skilled artisan would have been motivated to combine Thind’s teachings with the teachings of

Christiansen and Eylon. *See* Resp. 46–48; Paper 30, 2. In particular, Patent Owner contends that “the combination of Eylon and Christiansen already provides structural features that preserve ‘namespace consistency between selected objects maintained by a file system and a filter associated therewith.’” Resp. 46 (quoting Ex. 1008, 1:46–47, code (57)). Patent Owner explains that “[b]ecause each change to the virtual file system is implemented through the VFS itself, there is no need to maintain mainspace [sic] consistency as the commands to modify the contents of are instituted in serial order and that ordering of events inherently maintains namespace consistency.” *Id.* According to Patent Owner, Christiansen does not need “the deletion-on-close feature of Thind because Christiansen has its own mechanism for keeping the most up-to-date case copy of content on the file server.” Paper 27, 4 (emphases omitted); Paper 30, 2.

In addition, Patent Owner contends that the “complexity introduced” into the proposed combination “can only be realized by removing features from Eylon” and “[i]t is these very features that Eylon teaches as important.” Resp. 47–48.

Petitioner responds to Patent Owner’s contention concerning the “complexity introduced” into the proposed combination by asserting that Patent Owner “never identifies which features must be removed or explain[s] why they might need to be removed.” Reply 23.

For the reasons stated by Petitioner and supported by Dr. Houh’s testimony, we agree with Petitioner that an ordinarily skilled artisan would have been motivated to combine Thind’s extended-data functionality with Christiansen’s create operation. *See* Pet. 72–73; Ex. 1003 ¶¶ 380–385; Ex. 1006 ¶ 60; Ex. 1008, 1:40–44, 9:40–49, 12:13–14, 13:4–11, Fig. 12

(steps 1225, 1230, and 1235). In particular, Thind’s extended-data functionality would have provided “several advantages,” including (1) “preventing the filter metadata from being out of sync” and (2) “maintaining the ability of the filter to behave properly and perform its intended function.” *See* Pet. 73; Ex. 1003 ¶ 384. We also agree that an ordinarily skilled artisan would have had a reasonable expectation of success implementing the proposed combination. *See* Pet. 73; Ex. 1003 ¶¶ 383–384. Among other things, Thind and Christiansen name two common inventors and contain identical drawings as Figures 2–4 as well as identical descriptions of those drawings. *See* Ex. 1006 ¶¶ 32–42, code (75), Figs. 2–4; Ex. 1008, 5:49–7:22, code (75), Figs. 2–4.

As for Patent Owner’s contentions that “the combination of Eylon and Christiansen already provides structural features” that preserve namespace consistency and that Christiansen does not need “the deletion-on-close feature of Thind,” Thind’s extended-data functionality would have provided “several advantages” absent from the Christiansen-Eylon system, such as “preventing the filter metadata from being out of sync.” *See* Resp. 46; Paper 27, 4; Ex. 1003 ¶ 384; Ex. 1008, 1:40–44, 9:40–49, 13:4–11, Fig. 12 (steps 1225, 1230, and 1235). Those advantages would have motivated an ordinarily skilled artisan to combine Thind’s teachings with the teachings of Christiansen and Eylon. Ex. 1003 ¶ 384.

As for Patent Owner’s contention concerning the “complexity introduced” into the proposed combination, we agree with Petitioner that Patent Owner “never identifies which features must be removed or explain[s] why they might need to be removed.” *See* Resp. 47–48; Reply 23; Sur-reply 1–17.

4. CONCLUSION ABOUT OBVIOUSNESS/NONOBVIOUSNESS FOR CLAIM 13

For the reasons discussed above, the combined disclosures in Christiansen, Eylon, and Thind teach claim 13's subject matter. *See supra* §§ III.D.3(a)–(n), III.G.2. Moreover, an ordinarily skilled artisan would have been motivated to combine Thind's teachings with Christiansen's teachings as Petitioner proposes and would have had a reasonable expectation of success. *See supra* § III.G.3. Also, we accord little weight to Patent Owner's evidence concerning objective indicia of nonobviousness. *See supra* § III.D.3(p). Additionally, Patent Owner does not argue that any objective indicia of nonobviousness resulted directly from an allegedly unique characteristic of claim 13. *See Resp.* 53–56. Hence, Petitioner has shown by a preponderance of the evidence that claim 13 is unpatentable under § 103(a) as obvious over Christiansen, Eylon, and Thind.

IV. PETITIONER'S MOTION TO EXCLUDE AND STRIKE

Petitioner filed a motion to exclude certain exhibits and strike certain arguments. Paper 20. Petitioner moves to exclude as “new evidence” not authorized by 37 C.F.R. § 42.23(b) the following two exhibits submitted with the Sur-reply: Exhibit 2132, a Declaration of William Anthony Mason; and Exhibit 2133, a File Systems Filter Driver Design Guide. *See id.* at 1–2. Petitioner moves to strike arguments in the Sur-reply that allegedly “do not fairly respond to any argument in” the Reply, i.e., arguments in the Sur-reply relating to claim 1's preamble, limitation [1.b.i], limitation [1.c.i], and the motivation to combine Eylon's teachings with Christiansen's teachings. *See id.* at 1, 4–11.

Patent Owner responds to Petitioner's motion to exclude and strike by (1) withdrawing Exhibits 2132 and 2133 and (2) asserting that the arguments

in the Sur-reply properly respond to the arguments in the Reply. *See* Paper 22, 1, 4–14.

Due to Patent Owner’s withdrawal of Exhibits 2132 and 2133, Petitioner’s request to exclude those exhibits is moot. Additionally, we have considered the allegedly improper arguments in the Sur-reply, and they do not undermine Petitioner’s positions. Hence, we deny Petitioner’s motion to exclude and strike.

V. CONCLUSION

Based on the evidence presented with the Petition, the evidence introduced during the trial, and the parties’ respective arguments, Petitioner has shown by a preponderance of the evidence that:

- (1) claims 1–12 and 14–20 are unpatentable under § 103(a) as obvious over Christiansen and Eylon;
- (2) claim 2 is unpatentable under § 103(a) as obvious over Christiansen, Eylon, and Pudipeddi;
- (3) claim 12 is unpatentable under § 103(a) as obvious over Christiansen, Eylon, and Prahlad; and
- (4) claim 13 is unpatentable under § 103(a) as obvious over Christiansen, Eylon, and Thind.⁸

⁸ Should Patent Owner wish to pursue amendment of the challenged claims in a reissue or reexamination proceeding after the issuance of this Final Written Decision, we draw Patent Owner’s attention to the April 2019 Notice Regarding Options for Amendments by Patent Owner Through Reissue or Reexamination During a Pending AIA Trial Proceeding. *See* 84 Fed. Reg. 16,654 (Apr. 22, 2019). If Patent Owner chooses to file a reissue application or a request for reexamination of the challenged patent, we remind Patent Owner of his continuing obligation to notify the Board of any such related matters in updated mandatory notices. *See* 37 C.F.R. §§ 42.8(a)(3), (b)(2).

But Petitioner has not shown by a preponderance of the evidence that claim 13 is unpatentable under § 103(a) as obvious over Christiansen and Eylon.

In summary:

Claims	35 U.S.C. §	Reference(s)/ Basis	Claims Shown Unpatentable	Claims Not Shown Unpatentable
1–20	103(a)	Christiansen, Eylon	1–12, 14–20	13
2	103(a)	Christiansen, Eylon, Pudipeddi	2	
12	103(a)	Christiansen, Eylon, Pralhad	12	
13	103(a)	Christiansen, Eylon, Thind	13	
Overall Outcome			1–20	

VI. ORDER

Accordingly, it is

ORDERED that claims 1–20 in the '808 patent are determined to be unpatentable;

FURTHER ORDERED that Petitioner's motion to exclude and strike (Paper 20) is denied; and

FURTHER ORDERED that, because this is a Final Written Decision, the parties to the proceeding seeking judicial review of the decision must comply with the notice and service requirements of 37 C.F.R. § 90.2.

IPR2021-00015
Patent 8,380,808 B2

PETITIONER:

Joseph A. Micallef
Scott M. Border
Nathan A. Greenblatt
SIDLEY AUSTIN LLP
jmicallef@sidley.com
sborder@sidley.com
ngreenblatt@sidley.com
iprnotices@sidley.com

PATENT OWNER:

Mark Lawrence Lorbiecki
Brian G. Bodine
Daniel Andrew Brown
WILLIAMS, KASTNER & GIBBS PLLC
mlorbiecki@williamskastner.com