

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

UNIFIED PATENTS, LLC,
Petitioner,

v.

DATA CLOUD TECHNOLOGIES, LLC,
Patent Owner.

IPR2021-00361
Patent 6,560,613 B1

Before JONI Y. CHANG, NORMAN H. BEAMER, and
STEPHEN E. BELISLE, *Administrative Patent Judges*.

BELISLE, *Administrative Patent Judge*.

JUDGMENT
Final Written Decision
Determining All Challenged Claims Unpatentable
35 U.S.C. § 318(a)

I. INTRODUCTION

A. *Case Posture*

Unified Patents, LLC (“Petitioner”) filed a Petition (Paper 1, “Pet.”) requesting an *inter partes* review of claims 1, 2, 5, 7, 14, and 17 (“Challenged Claims”) of U.S. Patent No. 6,560,613 B1 (Ex. 1001, “the ’613 patent”). DataCloud Technologies, LLC (“Patent Owner”) filed a Preliminary Response to the Petition (Paper 7, “Prelim. Resp.”; see Paper 5, 1). We instituted an *inter partes* review of claims 1, 2, 5, 7, 14, and 17 of the ’613 patent on all grounds of unpatentability alleged in the Petition. Paper 8 (“Institution Decision” or “Dec.”).

After institution, Patent Owner filed a Response. Paper 12 (“PO Resp.”). Petitioner filed a Reply. Paper 14 (“Pet. Reply”). Patent Owner did not file a Sur-Reply. We held a hearing on April 27, 2022, and a transcript of the hearing appears in the record. Paper 19 (“Tr.”).

We have jurisdiction under 35 U.S.C. § 6. Under the applicable evidentiary standard, Petitioner has the burden to prove unpatentability by a preponderance of the evidence. See 35 U.S.C. § 316(e) (2018); 37 C.F.R. § 42.1(d) (2020). This Final Written Decision is issued pursuant to 35 U.S.C. § 318(a) and 37 C.F.R. § 42.73.

For the reasons discussed below, we determine Petitioner has established by a preponderance of the evidence that claims 1, 2, 5, 7, 14, and 17 of the ’613 patent are unpatentable.

B. *Related Proceedings*

The parties indicate that the ’613 patent is involved in five U.S. district court actions: *DataCloud Technologies, LLC v. Box, Inc.*, No. 1:20-cv-00763-LPS (D. Del.) (voluntarily dismissed on November 30, 2020);

DataCloud Technologies LLC v. F5 Networks, Inc., No. 2:20-cv-00872-DWC (W.D. Wa.) (voluntarily dismissed on September 15, 2020); *DataCloud Technologies LLC v. Extreme Networks, Inc.*, No. 1:20-cv-00764-LPS (D. Del.) (voluntarily dismissed on April 9, 2021); *DataCloud Technologies, LLC v. Wix.com, Inc.*, No. 1:20-cv-01314-LPS (D. Del.) (voluntarily dismissed on January 4, 2021); and *DataCloud Technologies, LLC v. 1&1 Ionos Inc.*, No. 1:20-cv-01313-LPS (D. Del.) (voluntarily dismissed on February 9, 2021). Pet. 63–64; Paper 5, 1.

Patent Owner also submits that, in addition to the above-cited actions, the following additional district court actions “may be affected by” a decision in this proceeding: *DataCloud Technologies, LLC v. Limelight Networks, Inc.*, No. 1:21-cv-00170 (D. Del.) (voluntarily dismissed on August 2, 2021); *DataCloud Technologies, LLC v. Squarespace, Inc.*, No. 1:21-cv-00164 (D. Del.) (pending); and *DataCloud Technologies, LLC v. Arista Networks, Inc.*, No. 1:21-cv-00155 (D. Del.) (voluntarily dismissed on May 5, 2021). Paper 5, 1.

Petitioner also submits, via its updated mandatory notices, that the ’613 patent also is involved in six additional district court actions: *DataCloud Technologies, LLC v. Check Point Software Technologies Inc.*, No. 1:21-cv-00837 (D. Del.) (voluntarily dismissed on September 13, 2021); *DataCloud Technologies, LLC v. Charter Communications, Inc.*, No. 6-21-cv-00662 (W.D. Tex.) (voluntarily dismissed on June 10, 2022); *DataCloud Technologies, LLC v. Newfold Digital, Inc.*, No. 1-21-cv-01020 (D. Del.) (voluntarily dismissed on October 27, 2021); *DataCloud Technologies, LLC v. Calix, Inc.*, No. 1-21-cv-01629 (D. Del.) (voluntarily dismissed on June 10, 2022); *DataCloud Technologies, LLC v. F-Secure Corporation*,

IPR2021-00361
Patent 6,560,613 B1

No. 6-21-cv-01211 (W.D. Tex.) (voluntarily dismissed on June 10, 2022); and *DataCloud Technologies, LLC v. Cequel Communications, LLC*, No. 6-21-cv-01275 (W.D. Tex.) (voluntarily dismissed on June 10, 2022).

Paper 13.

C. The '613 Patent

The '613 patent is titled “Disambiguating File Descriptors,” and issued on May 6, 2003, from U.S. Application No. 09/500,212, filed February 8, 2000. Ex. 1001, codes (10), (21), (22), (45), (54).

The '613 patent relates generally to “disambiguating file types on a computer system, and specifically to disambiguating communication channel file descriptors from file descriptors that . . . are associated with files stored on physical media.” Ex. 1001, 1:7–11. Figure 1 of the '613 patent is reproduced below.

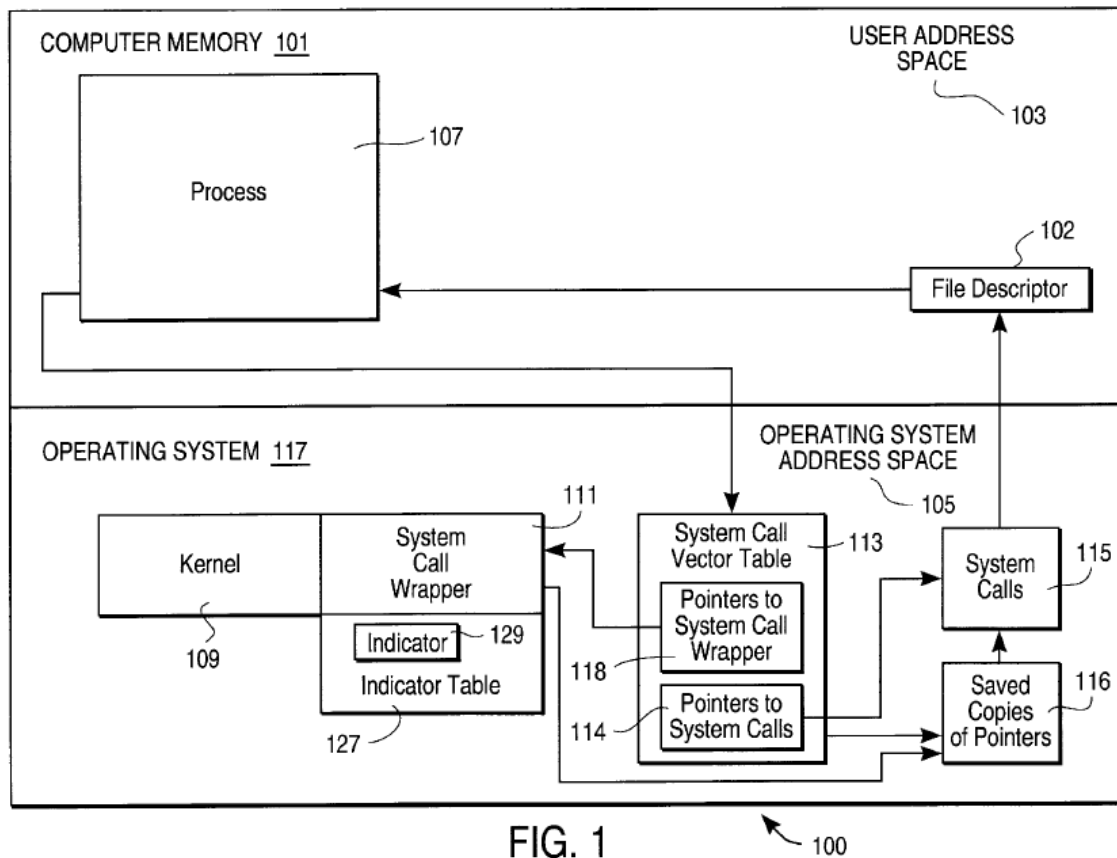


Figure 1 depicts a high-level block diagram illustrating a system for disambiguating file descriptors.

Id. at 6:65–67, Fig. 1.

Figure 1 shows system 100 for disambiguating file descriptors 102, including computer memory 101 having user address space 103 and operating system address space 105, wherein process 107 executes in user address space 103. *Ex.* 1001, 7:29–37. System 100 also includes operating system 117, which includes system call wrapper 111 and a data structure, such as indicator table 127, for storing indicators 129. *Id.* at 8:36–49. System call wrapper 111 maintains indicator table 127 so that indicator table 127 includes, for each file descriptor 102 that is associated with a specific file type, indicator 129 that indicates the association. *Id.* According

to the '613 patent, “the term ‘file type’ can be defined as broadly or narrowly as desired” (*id.* at 6:1–2), and may include communication channels (*id.* at 2:59–62), hardware devices and system resources (*id.* at 12:60–64), and files stored on media (*id.* at 12:64–65), and more narrowly, “files stored on media in specific directories, files with a specific attribute such as read only, and even a specific file (a file type of one)” (*id.* at 12:65–67).

In an exemplary embodiment, after a system call establishes a communication channel, the system call returns file descriptor 102 associated with the established communication channel, and this file descriptor is to be used by processes 107 to access the established communication channel. Ex. 1001, 9:30–42. After the system call terminates, system call wrapper 111 stores (in indicator table 127) indicator 129 that indicates that file descriptor 102 returned by the system call is associated with a communication channel. *Id.* at 9:42–46.

According to the '613 patent, because indicator table 127 includes indicators 129 for file descriptors associated with communication channels, “indicator table 127 can be utilized to determine whether or not a specific file descriptor 102 is associated with a communication channel.” Ex. 1001, 11:12–16. More specifically, “indicator table 127 can be examined to determine if an indicator [129] concerning the specific file descriptor 102 is *present*,” and “[i]f so, the file descriptor [102] is associated with a communication channel,” but “[i]f no such indicator [129] is present, the file descriptor 102 is not associated with a communication channel.” *Id.* at 11:16–21 (emphasis added).

Notably, although the '613 patent describes what “indicator 129” *does* (namely, indicates a file type associated with a file descriptor), neither party directs us to (and we do not find) any description in the '613 patent of what “indicator 129” *is* or of an example of “indicator 129.” See Ex. 1001, 8:36–49.

D. Illustrative Claim

The '613 patent includes 21 claims, of which claims 1, 2, 5, 7, 14, and 17 are challenged. Claims 1, 5, 14, and 17 are the challenged independent claims. Claim 1 is illustrative and reproduced below.

1. A method in a computer system for disambiguating file descriptors, the method comprising:
intercepting system calls that establish a file descriptor;
storing at least one indicator of a file type associated with an established file descriptor;
intercepting system calls that create a copy of at least one file descriptor;
storing at least one indicator concerning a created copy of a file descriptor; and
upon an attempt to access a file via a file descriptor, examining at least one stored indicator to determine with what file type the file descriptor is associated.

Ex. 1001, 14:46–58.

E. Evidence

Petitioner relies upon the following references:

Dukach, U.S. Patent No. 6,609,159 B1 (Ex. 1005, “Dukach”), filed November 30, 1998, and issued August 19, 2003.

M.R.M. Dunsmuir & G.J. Davies, Programming the UNIX System (1985) (Ex. 1007, “Dunsmuir”). Petitioner

submits Dunsmuir was “first published and publicly accessible in the British Library in 1985,” citing the Declaration of Sylvia D. Hall-Ellis, Ph.D. Pet. 28 (citing Ex. 1008 ¶¶ 40, 42–45, 51).
Pet. 22–28.

Petitioner also relies upon the Declaration of Philip Greenspun, Ph.D. (Ex. 1002).

Patent Owner did not submit any testimony from its own technical declarant, and did not cross-examine Petitioner’s declarant, Dr. Greenspun.

F. Asserted Grounds of Unpatentability

We instituted *inter partes* review of claims 1, 2, 5, 7, 14, and 17 of the ’613 patent on the following grounds asserted by Petitioner. Dec. 2–3, 36–37; Pet. 29.

Claims Challenged	35 U.S.C. §	Reference(s)
1, 2, 5, 7, 14, 17	102(e) ¹	Dukach
1, 2, 5, 7, 14, 17	103(a)	Dukach
1, 2, 5, 7, 14, 17	103(a)	Dukach, Dunsmuir

II. ANALYSIS

A. Applicable Law

Petitioner challenges the patentability of claims 1, 2, 5, 7, 14, and 17 of the ’613 patent on the grounds that the claims would have been anticipated under 35 U.S.C. § 102(e) or obvious under 35 U.S.C. § 103(a) in light of various references, namely, Dukach and Dunsmuir. To prevail in its challenges to the patentability of the claims, Petitioner must establish

¹ The Leahy-Smith America Invents Act (“AIA”), Pub. L. No. 112-29, 125 Stat. 284, 287–88 (2011), amended 35 U.S.C. §§ 102 and 103. Because the ’613 patent was filed before March 16, 2013, the effective date of the relevant amendment, the pre-AIA versions of §§ 102 and 103 apply.

unpatentability by a preponderance of the evidence. 35 U.S.C. § 316(e); 37 C.F.R. § 42.1(d). “In an [*inter partes* review], the petitioner has the burden from the onset to show with particularity why the patent it challenges is unpatentable.” *Harmonic Inc. v. Avid Tech., Inc.*, 815 F.3d 1356, 1363 (Fed. Cir. 2016) (citing 35 U.S.C. § 312(a)(3) (requiring *inter partes* review petitions to identify “with particularity . . . the evidence that supports the grounds for the challenge to each claim”)). This burden never shifts to Patent Owner. *See Dynamic Drinkware, LLC v. Nat’l Graphics, Inc.*, 800 F.3d 1375, 1378 (Fed. Cir. 2015) (citing *Tech. Licensing Corp. v. Videotek, Inc.*, 545 F.3d 1316, 1326–27 (Fed. Cir. 2008)) (discussing the burden of proof in *inter partes* review).

1. Anticipation

To serve as an anticipatory reference under 35 U.S.C. § 102, “the reference must disclose each and every element of the claimed invention, whether it does so explicitly or inherently.” *In re Gleave*, 560 F.3d 1331, 1334 (Fed. Cir. 2009). “The identical invention must be shown in as complete detail *as is contained in the . . . claim.*” *Richardson v. Suzuki Motor Co.*, 868 F.2d 1226, 1236 (Fed. Cir. 1989) (emphasis added). The elements must be arranged as required by the claim, “but this is not an ‘*ipsissimis verbis*’ test,” i.e., identity of terminology is not required. *In re Bond*, 910 F.2d 831, 832–33 (Fed. Cir. 1990) (citing *Akzo N.V. v. United States Int’l Trade Comm’n*, 808 F.2d 1471, 1479 & n.11 (Fed. Cir. 1986)).

2. Obviousness

A claim is unpatentable under 35 U.S.C. § 103 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.” *KSR Int’l Co. v. Teleflex Inc.*, 550 U.S. 398, 406 (2007). The question of obviousness is resolved on the basis of underlying factual determinations, including (1) the scope and content of the prior art; (2) any differences between the claimed subject matter and the prior art; (3) the level of skill in the art; and (4) when of record, objective evidence of non-obviousness, i.e., secondary considerations. *Graham v. John Deere Co.*, 383 U.S. 1, 17–18 (1966). Secondary considerations may include the following: “commercial success, long felt but unsolved needs, failure of others, etc.”² *Id.* The totality of the evidence submitted may show that the challenged claims would not have been obvious to one of ordinary skill in the art. *In re Piasecki*, 745 F.2d 1468, 1471–72 (Fed. Cir. 1984). When evaluating a combination of teachings, we must also “determine whether there was an apparent reason to combine the known elements in the fashion claimed by the patent at issue.” *KSR*, 550 U.S. at 418 (citing *In re Kahn*, 441 F.3d 977, 988 (Fed. Cir. 2006)).

The Supreme Court has made clear that we apply “an expansive and flexible approach” to the question of obviousness. *Id.* at 415. Whether a patent claiming a combination of prior art elements would have been obvious is determined by whether the improvement is more than the predictable use of prior art elements according to their established functions. *Id.* at 417. To reach this conclusion, however, requires more than a mere showing that the prior art includes separate references covering each separate limitation in a claim under examination. *Unigene Labs., Inc. v.*

² Patent Owner did not present any evidence or arguments directed to secondary considerations during this proceeding.

Apotex, Inc., 655 F.3d 1352, 1360 (Fed. Cir. 2011). Rather, obviousness requires the additional showing that a person of ordinary skill at the time of the invention would have selected and combined those prior art elements in the normal course of research and development to yield the claimed invention. *Id.* “To satisfy its burden of proving obviousness, a petitioner cannot employ mere conclusory statements. The petitioner must instead articulate specific reasoning, based on evidence of record, to support the legal conclusion of obviousness.” *In re Magnum Oil Tools Int’l, Ltd.*, 829 F.3d 1364, 1380 (Fed. Cir. 2016).

We analyze the challenges presented in the Petition in accordance with the above-stated principles.

B. Level of Ordinary Skill in the Art

Petitioner contends that a person of ordinary skill in the art, at the time of the invention of the ’613 patent, “would have had a Bachelor’s degree in computer science or a related field, and two years of work experience in operating systems, file systems, networks, and/or client-server applications, though additional education or work experience can make up for more or less formal education.” Pet. 20 (citing Ex. 1002 ¶ 31). Patent Owner does not present an alternative definition of the skilled artisan, and does not dispute Petitioner’s definition thereof. *See generally* PO Resp.

In determining the level of ordinary skill in the art, various factors may be considered, including the “type of problems encountered in the art; prior art solutions to those problems; rapidity with which innovations are made; sophistication of the technology; and educational level of active workers in the field.” *In re GPAC Inc.*, 57 F.3d 1573, 1579 (Fed. Cir. 1995) (citation omitted). The level of ordinary skill in the art also is reflected by

the prior art of record. *See Okajima v. Bourdeau*, 261 F.3d 1350, 1355 (Fed. Cir. 2001).

Considering the subject matter of the '613 patent, the background technical field, the prior art, and Petitioner's unopposed proposed definition of the skilled artisan, we determine (a) the skilled artisan here would have had a Bachelor's degree in computer science, or some equivalent level of education, and two years of work experience in computer operating systems, file systems, networks, or client-server applications; and (b) this would have provided a sufficient level of skill in light of the technology at issue in the '613 patent and prior art.

Regardless, neither party argues that the outcome of this case would differ based on our adoption of any particular definition of the level of ordinary skill in the art.

C. Claim Construction

We construe claims “using the same claim construction standard that would be used to construe the claim in a civil action under 35 U.S.C. 282(b), including construing the claim in accordance with the ordinary and customary meaning of such claim as understood by one of ordinary skill in the art and the prosecution history pertaining to the patent.” 37 C.F.R. § 42.100(b) (2020); *see also Phillips v. AWH Corp.*, 415 F.3d 1303 (Fed. Cir. 2005) (en banc).

In this context, claim terms “are generally given their ordinary and customary meaning” as understood by a person of ordinary skill in the art in question at the time of the invention. *Phillips*, 415 F.3d at 1312–13; *see CCS Fitness, Inc. v. Brunswick Corp.*, 288 F.3d 1359, 1366 (Fed. Cir. 2002) (There is “a ‘heavy presumption’ that a claim term carries its ordinary

and customary meaning.”). “In determining the meaning of the disputed claim limitation, we look 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). Extrinsic evidence is “less significant than the intrinsic record in determining ‘the legally operative meaning of claim language.’” *Phillips*, 415 F.3d at 1317.

Only those claim terms that are in controversy need to be construed, and only to the extent necessary to resolve the controversy. *See Nidec Motor Corp. v. Zhongshan Broad Ocean Motor Co.*, 868 F.3d 1013, 1017 (Fed. Cir. 2017) (stating that “we need only construe terms ‘that are in controversy, and only to the extent necessary to resolve the controversy’” (quoting *Vivid Techs., Inc. v. Am. Sci. & Eng’g, Inc.*, 200 F.3d 795, 803 (Fed. Cir. 1999))).

In the Petition, Petitioner submitted that the claim limitations in the ’613 patent should be given their ordinary and customary meaning. Pet. 20. In its Preliminary Response, Patent Owner had not proffered any proposed constructions for any claim limitations. *See generally* Prelim. Resp. Given the potentially dispositive nature of certain limitations in the Challenged Claims, in our Decision, we ordered the parties to submit proposed constructions for the terms “file,” “file descriptor,” and “indicator,” as recited, for example, in independent claim 1, and to the extent that a party proposed that such a term should have its ordinary and customary meaning, to submit and explicitly identify such meaning(s). Dec. 13, 37. To the extent necessary to resolve the controversy before us, we address claim

interpretation in our unpatentability analysis below. *See Nidec*, 868 F.3d at 1017.

D. Anticipation of Claims 1, 2, 5, 7, 14, and 17 by Dukach

Petitioner contends claims 1, 2, 5, 7, 14, and 17 are unpatentable under 35 U.S.C. § 102(e) as anticipated by Dukach (Ex. 1005).³ Pet. 22–63; Pet. Reply 12–24. Patent Owner opposes Petitioner’s contentions. PO Resp. 7–17. For the reasons expressed below, and based on the complete record before us, we determine that Petitioner has demonstrated by a preponderance of evidence that claims 1, 2, 5, 7, 14, and 17 are unpatentable as anticipated by Dukach. We turn first to an overview of Dukach.

1. Overview of Dukach (Ex. 1005)

Dukach generally relates to “client-server network computing, and, in particular, to methods, systems, and machine readable programming for interposing front end servers between servers and clients in such network computing,” as shown, for example, in Figure 1, reproduced below. Ex. 1005, 1:8–12, Fig. 1.

³ Dukach was not of record during prosecution of the ’613 patent. *See* Ex. 1001, code (56); Pet. 19–20 (“The grounds in this Petition rely on references not identified during any stage of prosecution of the ’613 patent.”).

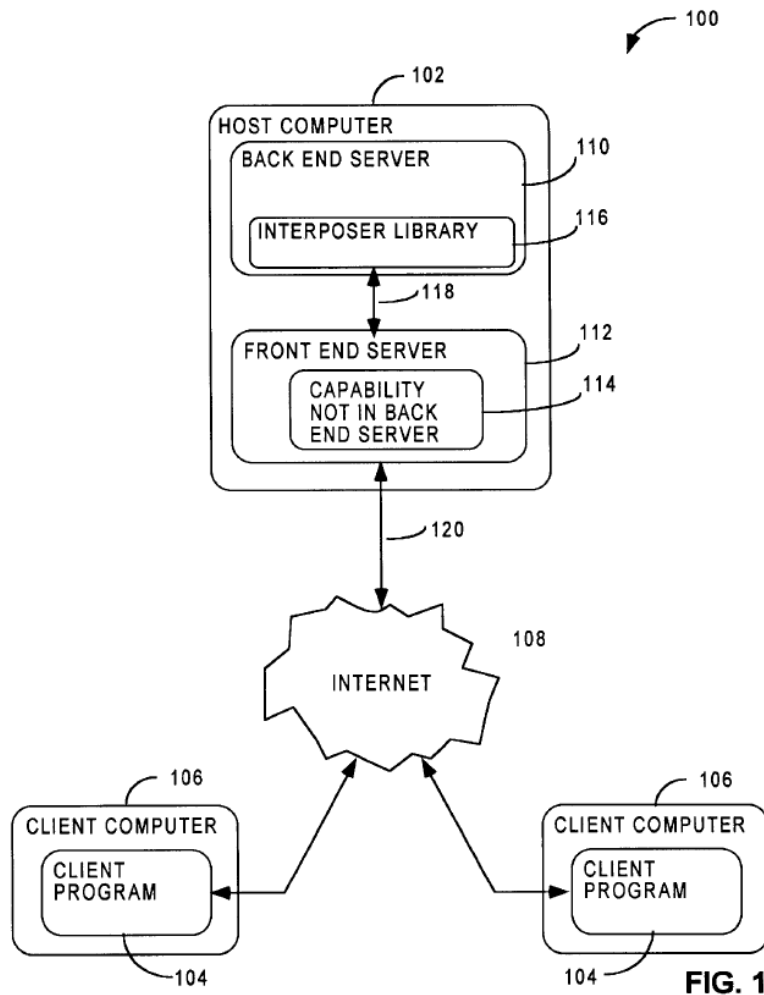


Figure 1 depicts a schematic overview of a client-server system including a host computer containing a back end server, an interposed library linked to the back end server, and an interposed front end server.

Id. at 5:13–17, Fig. 1.

Figure 1 shows client-server system 100 comprised of host computer 102 which communicates with a plurality of client programs 104, each located on client computer 106. Ex. 1005, 5:66–6:3. The computer network over which the server and clients communicate is the Internet 108, but could also be a local area network or a wide area network. *Id.* at 6:5–8. Host computer 102 also includes server program 110, which is called back

end server 110 to distinguish it from front end server 112 (which is loaded in the same operating system space on the same computer). *Id.* at 6:9–14.

Front end server 112 is designed to be interposed between back end server 110 and client computers 106. *Id.* at 6:14–16. Front end server 112 contains programming 114 “which provides extra functionality in the communication with clients not provided by the back end server.” *Id.* at 6:16–18. Back end server 110 works with front end server 112 via linking to interposed library 116. *Id.* at 6:18–20.

Dukach discloses “[t]he interposed library 116 is programmed to respond to certain back end OS [operating system] call[s] which it intercepts by communicating the information in those calls to the front end server through interprocess channels, or pipes [represented symbolically in Figure 1 by the arrow 118].” Ex. 1005, 6:29–34. Dukach also discloses:

[T]he interposed library includes functions 144A, such as bind(), listen(), and accept(), having some of the same names as the functions 144 contained in the OS’s network library 142. Since the interposed library is linked to the back end server with a higher precedence than the OS’s library, if the back end server calls a named OS function 144 for which there is a similarly named interposed library function 144A, the call will be intercepted by the interposed library function. This means the back end server process’s program control will go to the interposed library function 144A, rather than to the similarly named OS function 144.

Id. at 8:55–65. Front end server 112 “can send file descriptors of either direct connections with clients, or of pipes for relaying requests and response[s] from and to clients through the front end, to the interposed library for return to calls by [back end server 110] to accept connections.”

Id., Abstract. Front end server 112 can perform, for example, caching,

protocol conversion, session affinity, filtering, traffic conversion, and spreading load between multiple back end processes. *Id.*

Petitioner contends Dukach qualifies as prior art under 35 U.S.C. § 102(e) based on its filing date. Pet. 22. On this record, we have no evidence of an invention date other than the earliest possible effective filing date of the challenged claims. Thus, we determine that Dukach qualifies as prior art under 35 U.S.C. § 102(e) because Dukach's filing date of November 30, 1998, is before the earliest possible effective filing date of the challenged claims, which is February 8, 2000. Ex. 1001, code (22); Ex. 1005, code (22).

We further discuss below the disclosure of Dukach in connection with the parties' arguments.

2. Analysis

a) Independent Claim 1

Petitioner contends Dukach anticipates independent claim 1. Pet. 22–63. Patent Owner submits that “the crux of the dispute is whether or not Dukach stores [or examines] an indicator of a file type.” Tr. 20:24–21:1 (emphasis added); PO Resp. 3 (“[Petitioner] fails to meet its burden of establishing that the cited prior art discloses or teaches *storing and examining an indicator to determine a ‘file type.’*” (emphasis added)); see Pet. Reply 1 (“[Patent Owner’s] **sole** argument against unpatentability is that Dukach does not disclose [an] *indicator* [as] recited in the challenged claims.”), 12 (same). Patent Owner does not contend that the remaining limitations in claim 1 are absent in Dukach (*see generally* PO Resp.), and therefore, has waived such arguments. See 37 C.F.R. § 42.23(a) (“Any material fact not specifically denied may be considered admitted.”); *In re*

Nuvasive, Inc., 842 F.3d 1376, 1379–82 (Fed. Cir. 2016) (holding that patent owner waived arguments on an issue that were not raised in its response after institution); *see also Papst Licensing GmbH & Co. KG v. Samsung Elecs. Am., Inc.*, 924 F.3d 1243, 1250 (Fed. Cir. 2019) (holding patent owner forfeited argument for patentability not presented to the Board); *Bradium Techs. LLC v. Iancu*, 923 F.3d 1032, 1048 (Fed. Cir. 2019) (explaining that arguments not presented to the Board are waived). Nevertheless, for completeness, we address below all limitations of claim 1.

(1) “A method in a computer system for disambiguating file descriptors, the method comprising:”

The preamble of claim 1 recites a “method in a computer system for disambiguating file descriptors.” Ex. 1001, 14:46–47. Petitioner cites Dukach’s disclosure of computer system 102, which performs certain steps “to tell *whether* a file descriptor is associated with ‘a port of a service to be handled by the front end’ server *or not*.” Pet. 30 (citing Ex. 1005, 8:12–15, Fig. 1; Ex. 1002 ¶ 89); *see* Pet. 30–32 (citing, *inter alia*, Ex. 1005, 11:5–26, 11:36–40, 12:22–32, Fig. 1; Ex. 1002 ¶¶ 89–90). Neither Petitioner nor Patent Owner takes a position as to whether the preamble is limiting, nor does Patent Owner present arguments disputing the relied-upon Dukach disclosures. Pet. 30–32; PO Resp. 16–17 (Patent Owner arguing that “[e]ven assuming Dukach could be used for the same overall purpose [of disambiguating file descriptors], that does not mean the limitations of the challenged claims are present (or obvious).”); *see generally* PO Resp. 3–17. Because we find that Petitioner’s cited disclosure from Dukach discloses a “method in a computer system for disambiguating file descriptors,” we need

not determine whether the preamble is limiting. *See Nidec*, 868 F.3d at 1017.

(2) “intercepting system calls that establish a *file descriptor*”

Claim 1 recites “intercepting system calls that establish a *file descriptor*.” Ex. 1001, 14:48 (emphasis added). As discussed further below (see Section II.D.2.a.6), claim 1 also recites “access a *file* via a *file descriptor*.” *Id.* at 14:55 (emphases added). We turn first to the meanings of “file” and “file descriptor” as recited in claim 1.

(a) *Claim Construction: “file”*

The parties propose the following constructions for the term “file” as recited, for example, in claim 1:

Petitioner’s Proposed Construction	Patent Owner’s Proposed Construction
“any entity that is accessed via a <i>file descriptor</i> associated with the entity,” wherein “[t]he ‘entity’ here is not limited to only a file stored on media and can be other entities, such as communication channels or even hardware devices” (Pet. 20–22)	“a stream of data bytes that can be read and written (among other common operations)” (PO Resp. 3–4)

See Pet. Reply 2.

Petitioner submits that the term “file,” as recited, for example, in claim 1, would be understood by the skilled artisan to mean “any entity that is accessed via a file descriptor associated with the entity,” and that “[t]he ‘entity’ here is *not limited to* only a file stored on media and can be other entities, such as communication channels or even hardware devices.”

Pet. 20–21 (citing Ex. 1001, 3:3–5 (“[T]he system calls that access files via

file descriptors do not distinguish between a file on media and a communication channel.”), 3:13–15 (“Communication channels and files stored on media . . . both are accessed via file descriptors.”), 3:46–50 (“Under some operating systems, entities other than communication channels and files stored on media are treated as files, and hence accessed via file descriptors. For example, under the UNIX® operating system, hardware devices are designated by file descriptors.”), 5:66–6:13, 12:52–67, 13:28–37 (“In one embodiment the file type is *communication channels*. In another embodiment the file type is *files stored on media*. In yet another embodiment, the file type is *hardware devices*. Still other embodiments include other file types as desired.” (emphases added))); Ex. 1002 ¶ 78 (Petitioner’s declarant testifying to same); Pet. Reply 2–3 (citing, *inter alia*, Ex. 1001, 2:39–40 (“[A] system call . . . creates a file and returns a file descriptor that is associated with the newly created file. Subsequently, the file is accessed via the file descriptor.”)).

Patent Owner submits that the term “file” would be understood by the skilled artisan to have its “ordinary and customary meaning[],” allegedly “a stream of data bytes that can be read and written (among other common operations).” PO Resp. 3. But as noted by Petitioner (Pet. Reply 3–4), the only intrinsic evidence cited by Patent Owner does not support its proposed construction in the context of the ’613 patent; rather, such evidence favors Petitioner. See PO Resp. 4 (citing Ex. 1001, 2:35–45 (a “‘file is accessed via the file descriptor’”), 2:46–3:7 (“‘the system calls that access files via file descriptors do not distinguish between a file on media and a communication channel’”)); see also PO Resp. 7 (“It is undisputed that *file* descriptors were conventionally ‘associated with’ files stored on media, communication

channels, and even hardware devices.” (emphasis added)). Patent Owner also cites extrinsic evidence, namely, Dunsmuir (Ex. 1007), to support its proposed construction: “Conceptually, a file consists of a stream of data bytes followed by a logical end-of-file. ... Because UNIX file I/O is performed through a uniform file interface, a program that can read and write files can read and write anything.” PO Resp. 4 (quoting Ex. 1007, 22). But as noted by Petitioner (Pet. Reply 4–6), Patent Owner does not explain why this “conceptual” meaning set forth in the generic context of programming interfaces in a UNIX operating system would apply in the specific context of the ’613 patent. See Pet. Reply 4 (“[Patent Owner’s] proposed construction is problematic because it is inconsistent with the specification of the ’613 patent.”). Patent Owner also does not explain what “among other common operations” means in the context of its own proposed construction. For the foregoing reasons, and those discussed below, we agree with Petitioner’s proposed construction of “file.”

First, the claims of the ’613 patent support Petitioner’s proposed construction. For example, claim 1 recites that a “file” is accessed via a file descriptor. Ex. 1001, 14:55. Claim 5 recites a *file* descriptor being associated with “a communication channel” and determining with what *file* type the file descriptor is associated. Ex. 1001, 15:44–57. Together, these limitations show that a “file” is an entity assessed via a file descriptor, and such an entity includes, for example, a communication channel (i.e., per these claims, a communication channel is a type of *file* associated with a file descriptor). We find nothing in the claims, and Patent Owner does not direct us to anything in the claims, that supports limiting the term “file” only to a “stream of data bytes that can be read and written.”

Second, claims must be interpreted “‘in view of the specification’ *without unnecessarily importing limitations from the specification into the claims.*” *E-Pass Techs., Inc. v. 3Com Corp.*, 343 F.3d 1364, 1369 (Fed. Cir. 2003) (citing *Texas Digital Sys., Inc. v. Telegenix, Inc.*, 308 F.3d 1193, 1203–04 (Fed. Cir. 2002)) (emphasis added). Here, the Specification explicitly states that communication channels, files stored on media, hardware devices, and other file types are all specific types of “files”:

FIG. 4 illustrates a system 400 for utilizing an indicator table 127 to selectively intercept system calls 115 that access *files of a specific type* In one embodiment the file type is *communication channels*. In another embodiment the file type is *files stored on media*. In yet another embodiment, the file type is *hardware devices*. Still other embodiments include *other file types* as desired.

Ex. 1001, 13:28–37 (emphases added). Also, the Specification explicitly states that “[u]nder some operating systems, entities other than communication channels and files stored on media *are treated as files*, and hence accessed via file descriptors.” *Id.* at 3:46–50 (emphasis added). Thus, the Specification supports Petitioner’s proposed construction of “file.” We find nothing in the Specification, and Patent Owner does not direct us to anything in the Specification, that supports limiting the term “file” only to a “stream of data bytes that can be read and written.”

Third, turning to the prosecution history of the ’613 patent, neither party directs us with particularity to any statements in the prosecution history that inform this analysis.

Based on the foregoing, we construe “file” in the context of the ’613 patent to mean “any entity that is accessed via a file descriptor associated with the entity,” wherein “the ‘entity’ here is not limited to only a

file stored on media and can be other entities, such as communication channels or hardware devices.”

(b) *Claim Construction: “file descriptor”*

The parties propose the following constructions for the term “file descriptor” as recited, for example, in claim 1:

Petitioner’s Proposed Construction	Patent Owner’s Proposed Construction
“an integer that an operating system associates with, and uses to access, a <i>file</i> ” (Pet. Reply 7)	“an integer that the operating system associates with a given file” (PO Resp. 3)

See Pet. Reply 7.

Petitioner submits that the term “file descriptor” as recited, for example, in claim 1, would be understood by the skilled artisan to mean “an integer that an operating system associates with, and uses to access, a file.” Pet. Reply 7. More specifically, Petitioner argues “[a] file descriptor has a non-negative integer value, which is simply a *number*, and has an association with a ‘file.’” Pet. 2–3 (citing Ex. 1006, 397 (“A file is opened by the *open* system call, . . . [and t]his system call returns a *small integer*, called a *file descriptor*. This file descriptor may then be passed to a *read* or *write* system call (along with a *pointer* to a buffer and its size) to perform data transfers to or from the disk file or device.” (italics original))); Ex. 1002 ¶ 63.

Patent Owner submits that the term “file descriptor” would be understood by the skilled artisan to have its “ordinary and customary meaning[],” allegedly “an integer that the operating system associates with a given *file*.” PO Resp. 3. Patent Owner cites as support the same extrinsic evidence cited by Petitioner. *Id.* at 4 (citing Ex. 1006, 397 (“A file

descriptor is a small nonnegative integer ... used by the kernel to index an array of pointers for the current process ... to locate a file structure.”)); see Pet. 2–3.

Petitioner submits that its proposed construction “includes a requirement that the *descriptor* is used to access a file that is absent from [Patent Owner’s proposed construction],” but “[t]his difference is insubstantial as neither party contends this difference affects the unpatentability analysis.” Pet. Reply 7. “[T]o reduce unnecessary disputes, Petitioner concedes to [Patent Owner’s] construction of file descriptor.” *Id.*

Because the parties no longer dispute the meaning of “file descriptor,” and because neither party has directed us to any evidence contradicting that meaning, we adopt that meaning and construe “file descriptor” in the context of the ’613 patent to mean “an integer that the operating system associates with a given file.”

(c) *Subject Limitation*

Dukach discloses:

The interposed library’s dup() function 144AE intercepts back end server dup() calls to the OS. A dup() call is made with an original file descriptor to get a new alias file descriptor identifying the same socket as the original one. The interposed library intercepts dup() calls so that it can keep track of all the alias[es] which the back end has for any file descriptors associated with a network service being handled by the front end. Although not shown in the figures, *the interposed library also has similar functions which intercept dup2() and fcnt1() OS function calls, since these functions can also be used to obtain aliases for file descriptors.*

Ex. 1005, 16:15–26 (emphasis added). Petitioner contends the skilled artisan “would have understood that [the system call] dup2() creates, or

‘establishes,’ a new file descriptor (called *new_fd*) by giving it an integer value (called *new_value*) as its number and associating it with the entity that another file descriptor (called *old_fd*) is associated with: *new_fd* = *dup2(old_fd, new_value)*.” Pet. 33 (citing Ex. 1007, 65; Ex. 1002 ¶¶ 91); see Pet. 33–34 (explaining how “the *dup2()* call [newly] establishes the *new_fd* file descriptor as the call’s output”). Petitioner contends the foregoing description discloses “the interception of *dup2()* system calls that establish a file descriptor,” which Petitioner contends discloses “intercepting system calls that establish a file descriptor,” as recited in claim 1. Pet. 32–34 (citing Ex. 1002 ¶¶ 91–92).

Separately, Petitioner contends Dukach also discloses “the interception of *accept()* system calls that establish a file descriptor” (Pet. 34–35 (citing Ex. 1005, 12:57–59, 10:16–21; Ex. 1002 ¶¶ 93–94)), and “the *bind()* system calls are intercepted and that the intercepted *bind()* calls (*i.e.*, the alternative subroutines in lieu of the original *bind()* calls) establish a new file descriptor” (Pet. 35–37 (citing Ex. 1005, 10:61–62, 10:66–11:4, 11:13–35; Ex. 1002 ¶¶ 91–92, 95)). Petitioner contends that these descriptions of *accept()* and *bind()* system calls also each disclose “intercepting system calls that establish a file descriptor,” as recited in claim 1. Pet. 34–37 (citing Ex. 1002 ¶¶ 93–95).

Based on the foregoing evidence, Petitioner contends, and we find, Dukach discloses “intercepting system calls that establish a file descriptor,” as recited in claim 1. Patent Owner does not contest that this limitation is present in Dukach. See generally PO Resp. Any such argument has been waived by Patent Owner. See 37 C.F.R. § 42.23(a).

(3) “storing at least one indicator of a file type associated with an established file descriptor”

Claim 1 recites “storing at least one *indicator* of a file type associated with an established file descriptor.” Ex. 1001, 14:49–50 (emphasis added). This is the “crux” of this case—whether Dukach discloses storing (or examining) the claimed “indicator.” See, e.g., Tr. 20:24–21:1. We turn first to the meaning of “indicator” as recited in claim 1.

(a) *Claim Construction: “indicator”*

In our Institution Decision, we noted that “although the ’613 patent describes what ‘indicator 129’ *does* (namely, indicates a file type associated with a file descriptor), neither party directs us to (and we do not find) any description in the ’613 patent of what ‘indicator 129’ *is* or of an example of ‘indicator 129.’” Dec. 6 (citing Ex. 1001, 8:36–49). Despite subsequent additional briefing and oral argument in this case, the parties still appear to struggle to provide a supported definition for this critical term.

The parties propose the following constructions for the term “indicator” as recited, for example, in claim 1:

Petitioner’s Proposed Construction	Patent Owner’s Proposed Construction
“information or data that points to something” (Pet. Reply 7)	“a discrete piece of information associating a file descriptor and a file type” (PO Resp. 4)

See Pet. Reply 7.

Petitioner submits that the term “indicator,” as recited, for example, in claim 1, would be understood by the skilled artisan to mean “information or data that points to something.” Pet. Reply 7. As support, Petitioner cites to

a dictionary that defines “indicator” as “a thing that indicates or points to something.” *Id.* at 7–8 (citing Ex. 1016, 4).

Patent Owner submits that the term “indicator” would be understood by the skilled artisan to mean “a discrete piece of information associating a file descriptor and a file type.” PO Resp. 4; *see id.* at 5–6 (discussing what the “indicator” *does*). Patent Owner states: “While the Board correctly noted in its [Institution Decision] that the ’613 patent does not provide an explicit “description” or “example” of an *indicator* (Paper 8, 28), [the skilled artisan] would nevertheless have understood, in the context of the patent’s description and the [skilled artisan’s] experience, that an indicator is a discrete piece of information or data.” *Id.* at 6. During oral hearing, we asked Patent Owner’s counsel, “[W]hat is an indicator?” Tr. 22:11–13. Counsel’s response highlights the difficulty in defining “indicator” as anything more than information or data that indicates (or points to) something:

So I -- an indicator, in my experience, and I was a software developer, I have a slight bit of POSA knowledge, and the record evidence here is not that there is any -- *I don’t think there is any particular definition of an indicator that is suggested by the evidence, the prior art or, frankly, the testimony.*

You know, I do take -- while I wouldn’t say I concede to or agree to petitioner’s arguments in its reply brief about its definition that it’s an information or data that points to something, you know our position is that it’s a -- when we said discrete piece of information, I think of it as an affirmative indicator. In the institution decision I think there was a comment from the board that said something about, well, patent owner only argues about what it does, not what it is -- what an indicator does, not what it is. Respectfully, the claim itself requires that the indicator does something. *The indicator indicates something. It is data.* It could be a data structure. It could be -- it can be a number, right?

I mean I think, frankly, a file descriptor itself indicates the number of the file descriptor. *It's a pointer.* That's what it effectively is, a file descriptor is a pointer to a file, right. So it indicates a file. But it indicates a file, it does not indicate information about the file. It is not an indication of the file type, and that's what the claim requires.

So even if you were to disagree with our explanation in the response about what an indicator is, that it's a discrete piece of information associating a file descriptor with a file type, *that association and that requirement is still in the claims themselves.* So even if you were to say that it's just information, it's just data, the claim still requires that it is *information or data about the file type*, not just information in the abstract. *It has to indicate something* as required by the claim and as taught by the specification.

Tr. 22:14–23:23 (emphases added).

We then asked Patent Owner's counsel, "[C]an you give me an example, even if it's not in the patent, of an indicator that indicates file type of a file? What is it?" Tr. 26:9–11. Counsel's response further highlights the claim construction challenge before us:

Well, I think that a person of ordinary skill in the art would understand in the same way that a person of ordinary skill in the art would understand from this -- this is not -- *I will concede that what I'm saying here is not strictly within the record*, but my argument would be that it's not because it doesn't need to be. Sort of like some of the arguments about file. File is not described or defined in the patent because everybody knows what a file is, and a file in fact is typically implemented as a really complicated data structure. Okay. *So an indicator could be a piece of data. It could be a data structure. Maybe it's a flag. I don't know what it is.* But it can -- I think that is representative of the flexibility of and not prescribing a particular implementation of what the indicator -- how it is specifically implemented, but a person of ordinary skill in the art would know that you could implement an indicator in various ways such as by a data structure like a structure that has various fields. *Maybe*

it's a table that maps, you know, communications to a value. Maybe it has a hardware device as a different value. Something like that.

Tr. 26:12–27:9 (emphases added).

Petitioner disputes that the meaning of “indicator” should itself require “associating a file descriptor and a file type.” Pet. Reply 8. In particular, Petitioner argues “other portions of the claim language following the term *indicator* expressly recite and describe the association between a file descriptor and a file type,” and therefore, “[Patent Owner’s] proposed construction of *indicator* as including ‘information associating a file descriptor and a file type’ renders the other terms of the claims superfluous.” *Id.* at 8–9 (citing case law regarding rendering claim language superfluous). Patent Owner did not file a sur-reply to respond to Petitioner’s arguments here. We agree with Petitioner that claim 1, for example, itself recites what the “indicator” is indicating, and thus a proper construction of “indicator” would not itself require “associating a file descriptor and a file type” as proffered by Patent Owner. We also find this better aligns with Patent Owner’s agreement that an indicator “*indicates something*” and “[i]s a *pointer*.” Tr. 23:5, 9.

Petitioner also disputes that the meaning of “indicator” should require “a *discrete* piece of” information that points to something, as opposed to any “information.” Pet. Reply 9–11. Petitioner argues, and we agree, “[Patent Owner] does not explain what ‘discrete’ means, nor does it point to any supporting evidence.” *Id.* at 9–10. For example, Patent Owner does not explain the difference between “information” and “*discrete* information,” let alone cite to any evidence supporting such a distinction (even if one exists). Petitioner also argues, and we agree, “[t]he ’613 patent specification does

not restrict an *indicator* to have a specific [sic] type or form of information, nor does it exclude any specific type or form of information.” *Id.* at 10; see *Thorner v. Sony Computer Entm’t Am. LLC*, 669 F.3d 1362, 1365 (Fed. Cir. 2012) (To act as its own lexicographer, a patentee must “clearly set forth a definition of the disputed claim term,” and “‘clearly express an intent’ to redefine the term.”); *In re Paulsen*, 30 F.3d 1475, 1480 (Fed. Cir. 1994) (holding that an inventor may define specific terms used to describe an invention, but must do so “with reasonable clarity, deliberateness, and precision” and, if done, “‘must set out his uncommon definition in some manner within the patent disclosure’ so as to give one of ordinary skill in the art notice of the change” in meaning) (quoting *Intellicall, Inc. v. Phonometrics, Inc.*, 952 F.2d 1384, 1387–88 (Fed. Cir. 1992)); *SciMed Life Sys., Inc. v. Advanced Cardiovascular Sys., Inc.*, 242 F.3d 1337, 1341 (Fed. Cir. 2001) (Disavowal requires that “the specification [or prosecution history] makes clear that the invention does not include a particular feature.”).

Based on the foregoing, we construe “indicator” in the context of the ’613 patent to mean “information or data that points to something.”

(b) *Subject Limitation*

Dukach discloses:

Once the control pipe 192 has been created, *step 178 calls the OS function dup2() to give the newly created control pipe the same file descriptor as that used by the back end server in the intercepted bind() call.* This is done so that when the back end server makes future calls using what it thinks is the file descriptor of a network service listen socket, the library can use that file descriptor to directly access the control pipe which the library has created for the network service.

Next, step 180 *adds the bind() call's file descriptor to list of monitored file descriptors*, so that when the library intercepts a future OS call with that file descriptor, it will know that the intercepted call is for *a service to be handled by the library and front end rather than directly by the OS*.

Ex. 1005, 11:27–41 (emphases added). Petitioner argues the skilled artisan “would have understood that the number of the bind() call's file descriptor is the same as that of the newly established file descriptor returned by dup2() . . . , which now is associated with the newly created control pipe 192 (i.e., a communication channel).” Pet. 39 (citing Ex. 1005, 11:27–35; Ex. 1002 ¶ 96). Petitioner contends the foregoing description discloses “after interception of the bind() calls, adding (or ‘storing’) the number of an established file descriptor (i.e., used as its ‘indicator’) into a list of monitored file descriptors associated with services to be handled by the front end server via a control pipe (which are a ‘file type’).” Pet. 38.

More specifically, Petitioner contends “Dukach utilizes the *presence or absence* of a file descriptor's number stored in a list of monitored file descriptors to *indicate* the file type associated with the file descriptor, which is how the '613 patent describes indicating a file type by the indicator's presence or absence in an indicator table.” Pet. 37–38 (internal citations omitted) (emphases added); see Ex. 1001, 11:12–42 (“The indicator table 127 can be examined to determine if an indicator 205 concerning the specific file descriptor 102 is *present*. If so, the file descriptor 201 is associated with a communication channel. If no such indicator 205 is present, the file descriptor 102 is not associated with a communication channel.” (emphasis added)). Petitioner contends Dukach's “list of the monitored file descriptors stores only those associated with services to be handled by the front end server, i.e., a ‘file type.’” Pet. 40 (citing Ex. 1005,

11:5–40). Simply put, Petitioner contends “the file descriptor number by its presence in the list of monitored file descriptor[s] is the claimed ‘indicator,’” and that in the foregoing context, “the ‘file type’ is associated with the control pipe, *i.e.*, a *communication channel for services to be handled by the front end server*, consistent with the description in the ’613 patent.” Pet. 40 (citing Ex. 1001, 11:12–42; Ex. 1002 ¶ 96).

Based on the foregoing, Petitioner contends, and we are persuaded, that Dukach discloses “storing at least one indicator of a file type associated with an established file descriptor,” as recited in claim 1. Pet. 37–40 (citing Ex. 1002 ¶¶ 96–97).

Separately, Petitioner also contends:

Dukach discloses that, similar to the `dup( )` system call, the `dup2( )` system call made by the back end server will be intercepted, and the intercepted `dup2( )` call will record the newly established file descriptor as follows:

- intercepted `dup2(old_fd, new_value)`
 - if the number of `old_fd` is on the list of monitored file descriptors
 - call `new_fd = dup2(old_fd, new_value)`
 - record `new_value` in the list of monitored file descriptors
 - return `new_fd` in standard OS format
 - else pass call to OS

Pet. 41 (citing Ex. 1005, 16:15–25, Fig. 12B; Ex. 1002 ¶ 98; Ex. 1007, 65). Petitioner contends the skilled artisan “would have understood, the newly established file descriptor `new_fd` will have the number value of `new_value`, which will be recorded (*i.e.*, ‘stor[ed]’), if the number of the file descriptor `old_fd` is on the list of monitored file descriptors.” Pet. 41. Petitioner

contends, and we are persuaded, that Dukach discloses, “after the interception of the dup2() calls” in the foregoing context, “storing at least one indicator of a file type associated with an established file descriptor,” as recited in claim 1. Pet. 40–41 (citing Ex. 1002 ¶ 98; Ex. 1007, 64).

In its Response, as in its Preliminary Response, Patent Owner contends the Petition fails to identify any prior art that discloses or teaches “storing and examining an indicator to determine a ‘file type.’” PO Resp. 3; see PO Resp. at 7–17; Prelim. Resp. 5 (alleging the asserted prior art fails to disclose or teach “storing an indicator of a file’s type or examining an indicator to determine what type of file a given file descriptor is associated with”), 3–12.

More specifically, Patent Owner asserts repeatedly that the Petition “relies on the file descriptor itself, which is ‘just a number’ and not an indicator of a file type.” PO Resp. 9; see, e.g., PO Resp. 10 (“[A]s [Petitioner] concedes, a file descriptor ‘is just a number—a non-negative integer—and thus does *not provide any information* on what ‘type’ of file it is associated with.”; “[Petitioner] essentially argues that the same file descriptor number that [Petitioner] admitted ‘does not provide any information’ about file type may be used *as* the indicator of the file type.”). This is the same argument Patent Owner made in its Preliminary Response. See, e.g., Prelim. Resp. 5–6, 10–12. But as we stated in our Institution Decision, and maintain here, it appears Patent Owner misconstrues Petitioner’s argument concerning the claimed “indicator.” See Dec. 27–31; Pet. Reply 1 (“[T]he Board had addressed at length and rejected this argument. . . . [Patent Owner’s Response] fails to present any new or

additional evidence to alter the Board’s Institution Decision.”). We find Patent Owner’s argument here unavailing.

Petitioner does not contend that a file descriptor’s number, itself and in the abstract, discloses the claimed “indicator,” but rather the inclusion (or storage or presence) of that number in a list of monitored file descriptors discloses such a stored “indicator.” *See, e.g.*, Pet. 49 (“[The skilled artisan] would have understood that, before the listen() call or the accept() call attempts to access a file using a file descriptor, it examines the file descriptor to determine *if it is on the list of monitored file descriptors*—the ‘indicators’ of a file type associated with the control pipe for services to be performed by the front end server.” (emphasis added)); Pet. 37–38 (“Dukach utilizes the *presence or absence* of a file descriptor’s number stored in a list of monitored file descriptors to *indicate* the file type associated with the file descriptor, which is how the ’613 patent describes indicating a file type by the indicator’s presence or absence in an indicator table.”); *see also* Pet. Reply 13 (“Petitioner contended in the Petition and maintains that it is the presence of a file descriptor’s number on the list of monitored file descriptors, not the file descriptor’s number itself, that teaches the claimed *indicator*.” (bold omitted)); *cf.* PO Resp. 9 (“[A]n ‘indicator’ . . . must reflect . . . an *indication* that such file descriptor is associated with a communication channel.”).

Patent Owner also variously argues “Dukach does not disclose or teach an ‘indicator’ of a ‘file type’ as claimed and disclosed in the ’613 patent,” or storing that indicator of a file type, but, notably, Patent Owner does so divorced from the actual, broad limitations of claim 1, *using only attorney-argument*, and still without giving us a single, concrete

example of what an “indicator” is (deficiencies we previously noted in its Preliminary Response (see Dec. 28–29)). PO Resp. 11, 13; *see generally id.* at 11–15. More specifically, Patent Owner argues “Dukach’s ‘monitored file descriptors,’ . . . were all of one type—network communications—and Dukach does not disclose, teach, or even mention storing an indicator of the type of ‘file’ associated with its file descriptors,” and “Dukach [does not] teach disambiguating between different types of ‘files’ using an ‘indicator of file type.’” PO Resp. 12. Patent Owner argues that Petitioner “improperly equate[s] the ‘file type’ required by the ’613 patent claims with Dukach’s monitored network services.” PO Resp. 12. Patent Owner argues the Petition “hinges on the false equivalence between information about the ‘type’ of file that is associated with a given file descriptor in the ’613 patent and the list of network services that Dukach chooses to have handled by its front end server.” PO Resp. 13. Patent Owner argues, in conclusory fashion, “Dukach does not use the presence or absence of that file descriptor to *distinguish* between file types. Nor does Dukach interpret the presence or absence of the file descriptor in the list in any way.” PO Resp. 15–16. We also find Patent Owner’s arguments here unavailing, in view of Petitioner’s explanation of how Dukach discloses the subject limitation (*see supra*), and as further discussed below.

For example, as asserted by Petitioner, the ’613 patent explicitly discloses, *inter alia*, disambiguating file descriptors associated with only two types of files—either a file stored on media or a communication channel (another type of file)—based on the presence or absence of an “indicator” of *one* type in an indicator table:

In some operating systems 117, every file descriptor 102 is either associated with a file stored on media or with a communication channel. In embodiments of the present invention that execute under such operating systems 117, the indicator table 127 can be utilized to determine *whether or not* any specific file descriptor 102 is associated with a communication channel, and *whether or not* any specific file descriptor 102 is associated with a file stored on media. The indicator table 127 can be utilized to determine whether or not a file descriptor 102 is associated with a communication channel, as described above. *Because file descriptors 102 that are not associated with communication channels are associated with files stored on media, the indicator table 127 can also be utilized to determine whether or not a specific file descriptor 102 is associated with a file stored on media.* In order to so determine, the indicator table 127 is examined to determine *if an indicator 205 concerning the specific file descriptor 102 is present.* If not, the file descriptor 102 is associated with a file stored on media. If such an indicator 205 is present, the file descriptor 102 is not associated with a file stored on media [but rather with a communication channel].

Ex. 1001, 11:12–42 (emphases added). Petitioner’s declarant, Dr. Greenspun, testifies that Dukach likewise discloses disambiguating file descriptors based on the presence or absence of a stored “indicator” of one file type in an indicator table:

The ‘613 Patent teaches using presence within a table of files of a given type as an indicator that a file descriptor is associated with a file of a given type. *cf.* ‘613 Patent, 11:12-42 (“The indicator table 127 can be examined to determine if an indicator 205 concerning the specific file descriptor 102 is **present**. If so, the file descriptor 201 is associated with a communication channel.”) Dukach describes a similar technique, looking for the presence or absence of a file descriptor’s number within a list of monitored file descriptors to determine whether a service associated with the file descriptor is to be handled by the front end server. . . . Here, the “file type” is associated with the control pipe, *i.e.*, a **communication channel**

for **services to be handled by the front end server.**
(*Cf.* '613 Patent, 11:12-42.)

Ex. 1002 ¶ 96; *see* Pet. 38–39, 47 (citing Ex. 1002 ¶¶ 96, 103).

In our Institution Decision, we credited the foregoing and related testimony of Dr. Greenspun (Ex. 1002 ¶¶ 96–98), and noted that Patent Owner at that stage of the proceeding had not offered any evidence to rebut Dr. Greenspun's testimony. Dec. 30–31. Despite that, as highlighted by Petitioner, Patent Owner still offers *no evidence* to rebut Dr. Greenspun's testimony in this regard. *See* Pet. Reply 16 (“But in arguing all these [above arguments], [Petitioner] presents no rebuttal evidence—neither a deposition testimony of Dr. Greenspun, nor a rebuttal testimony by its own expert, nor any other evidence.”); *see generally* PO Resp. Instead, Patent Owner merely declares Dr. Greenspun's testimony “conclusory” and tainted by “vague leap[s]” (*see, e.g.*, PO Resp. 14), but yet makes such accusations against a technical declarant with nothing but attorney-argument. Regardless, for the same reasons given by Petitioner (*see* Pet. Reply 16–18), we disagree with Patent Owner and continue to credit Dr. Greenspun's technical testimony as to the subject limitation. *See* Pet. Reply 16–18 (Petitioner explaining why “[Patent Owner's] attorney argument is wrong”).

As noted above, Patent Owner variously argues Dukach does not disclose “storing” of the indicator at issue in this limitation. *See generally* PO Resp. 7–17. To summarize, we disagree with Patent Owner, and as explained above, are persuaded by Petitioner's evidence, including the unrebutted testimony of Dr. Greenspun, that Dukach discloses storing information that points to a file type associated with a file descriptor. First, we are persuaded that, in Dukach, the *presence or absence* of a file descriptor's number *in the list of monitored file descriptors* constitutes

information that points to a file type associated with the file descriptor. See Pet. 37–40 (citing, *inter alia*, Ex. 1005, 11:5–40; Ex. 1002 ¶ 96; Ex. 1001, 11:12–42). Second, we also are persuaded that, in Dukach, the list of monitored file descriptors “needs to be maintained in memory such that it can be used by software to check an at-issue file descriptor’s presence or absence in the list,” which constitutes *storing* the indicator. Pet. Reply 16–17 (quoting Ex. 1002 ¶ 97); see Pet. 40 (citing Ex. 1002 ¶ 97). Finally, even if Patent Owner were correct that Dukach does not disclose a particular *embodiment* in the ’613 patent (see, e.g., PO Resp. 15 (discussing an embodiment “only applicable for binary file types”)), the issue before us is not whether Dukach indeed discloses such an embodiment, but rather whether Dukach discloses the invention *recited*, for example, in claim 1, given the breadth (scope) of the limitations defining that invention.

Based on the foregoing evidence, Petitioner contends, and we find, Dukach discloses “storing at least one indicator of a file type associated with an established file descriptor,” as recited in claim 1.

(4) “*intercepting system calls that create a copy of at least one file descriptor*”

Dukach discloses that dup() system calls are intercepted as follows:

The interposed library’s dup() function 144AE *intercepts back end server dup() calls to the OS.*

* * *

If the back end has called dup() with a monitored file descriptor, step 310 causes steps 312–316 to be performed. *Step 312 makes an OS dup() call for the file descriptor. Step 314 records the alias file descriptor returned by the OS in the list of monitored file descriptors as a duplicate associated with the file descriptor for which dup() was called.*

Ex. 1005, 16:15–16, 33–39; *cf.* Ex. 1001, 10:19–24 (“Some system calls 115 simply copy an existing file descriptor 102, without copying its associated file.”). Petitioner contends the skilled artisan “would have understood that dup() system calls ‘create a copy of a least one file descriptor.’” Pet. 41–42 (citing Ex. 1002 ¶ 99; Ex. 1007, 64). Petitioner contends the foregoing description discloses “intercepting system calls that create a copy of at least one file descriptor,” as recited in claim 1. Pet. 41–42 (citing Ex. 1002 ¶ 99).

Dukach also discloses:

The interposed library’s fork() function 144AF shown in FIG. 12B, *intercepts* fork() calls made by the back end server, such as those discussed above with regard to step 156 of FIG. 11. After a process makes a fork() call a new, child, process is created which has *copies of all the data structures, variables and file descriptors* possessed by the parent process which made the fork() call, exactly as they existed at the time of that call. The only difference between the two processes is that fork() returns to the child process with an indication that it is the child process, and to the original process with an indication that it is the parent process.

Ex. 1005, 16:43–53 (emphases added). Petitioner contends this description discloses “interception of the fork() call,” and that “the fork() call ‘makes . . . copies of all . . . file descriptors’ and thus creates a copy of at least one file descriptor.” Pet. 42–43 (citing Ex. 1005, 16:43–53; Ex. 1002, ¶ 100; Ex. 1007, 97). Petitioner contends the foregoing description also discloses “intercepting system calls that create a copy of at least one file descriptor,” as recited in claim 1. Pet. 42–43 (citing Ex. 1002 ¶ 100).

Based on the foregoing evidence, Petitioner contends, and we find, Dukach discloses “intercepting system calls that create a copy of at least one file descriptor,” as recited in claim 1. Patent Owner does not contest that

this limitation is present in Dukach. See generally PO Resp. Any such argument has been waived by Patent Owner. See 37 C.F.R. § 42.23(a).

(5) “storing at least one indicator concerning a created copy of a file descriptor”

Dukach discloses:

If the back end has called dup() with a monitored file descriptor, step 310 causes steps 312–316 to be performed. Step 312 makes an OS dup() call for the file descriptor. *Step 314 records the alias file descriptor returned by the OS in the list of monitored file descriptors as a duplicate associated with the file descriptor for which dup() was called.*

Ex. 1005, 16:33–39. Petitioner contends that this description discloses that, “after intercepting the dup() calls,” as discussed above, “the number of the file descriptor’s copy (the non-negative integer of the alias file descriptor established by the dup() call made inside the intercepted dup() call) is recorded in the list of monitored file descriptors.” Pet. 43–44 (citing Ex. 1002 ¶ 101); see Ex. 1005, Fig. 12B (“record new file descriptor returned by dup in list of monitored file descriptors, as dup of call’s file descriptor”). Petitioner contends the skilled artisan “would have understood that, ‘record[ing] the alias file descriptor[’s number]’ is substantially the same as ‘storing at least one indicator,’ and ‘a duplicate associated with the file descriptor’ is substantially the same as ‘a created copy of a file descriptor.’” Pet. 44–45 (citing Ex. 1005, 16:36–39; Ex. 1002 ¶ 101). Petitioner contends the skilled artisan would also understand that “a duplicate of a file descriptor made by the dup() calls is basically just a copy of the file descriptor that is created by dup().” Pet. 45 (citing Ex. 1002 ¶ 101).

Based on the foregoing, Petitioner contends Dukach discloses “storing at least one indicator concerning a created copy of a file descriptor,” as recited in claim 1. Pet. 43–45 (citing Ex. 1002 ¶ 101).

Separately, Petitioner contends “that Dukach’s fork() call is also ‘storing at least one indicator concerning a created copy of a file descriptor’ as claimed.” Pet. 45 (citing Ex. 1002 ¶¶ 100, 102). Petitioner argues:

[T]he intercepted fork() call, by calling fork() in step 320 after interception, creates copies of *everything* in the parent back end server, including its interposed library with a list of monitored file descriptors. [The skilled artisan] would have understood that the monitored file descriptors on the copied list are “indicators,” now “concerning” *the copies of the file descriptors in the child process* to indicate their association with services to be handled by the front end server.

Pet. 45 (internal citations omitted) (citing Ex. 1005, Fig. 12B; Ex. 1007, 97; Ex. 1002 ¶ 100). Petitioner contends “the child process’s monitored list of file descriptors, which are the indicators indicating the file descriptors’ association with services to be handled by the front end server, is also stored in the host computer’s memory.” Pet. 46 (citing Ex. 1002 ¶ 100). In view of the foregoing, Petitioner contends that Dukach’s fork() call also discloses “storing at least one indicator concerning a created copy of a file descriptor,” as recited in claim 1. Pet. 45–46 (citing Ex. 1002 ¶ 100).

Based on the foregoing evidence, Petitioner contends, and we find, Dukach discloses “storing at least one indicator concerning a created copy of a file descriptor,” as recited in claim 1. Except as discussed above concerning the limitation “storing at least one indicator of a file type associated with an established file descriptor” (see Section II.D.2.a.3, *supra*), Patent Owner does not otherwise contest that this limitation is present in

Dukach. See generally PO Resp. Any such argument has been waived by Patent Owner. See 37 C.F.R. § 42.23(a).

(6) “upon an attempt to access a file via a file descriptor, examining at least one stored indicator to determine with what file type the file descriptor is associated”

Dukach discloses:

When the back end calls listen() in step 154 of FIG. 11, that call will be intercepted by the interposed library listen() function 144AB of FIG. 12A, causing steps 210–220 to be executed.

Step 210 tests to see if the call’s file descriptor is on the interposed library’s list of monitored file descriptors, discussed above with regard to step 180 of the library’s bind() function. If not, the listen() call is for a service not being handled by the front end, and step 210 causes program flow to skip to step 220 which ends the execution of the library’s listen() function by passing the listen call directly to the OS.

If, however, the listen() call’s file descriptor is one of the library’s monitored file descriptors, meaning that it is for a network service handled by the front end, steps 212–218 are executed.

Step 212 writes a listen message, with the intercepted listen() call’s parameters, to file descriptor’s associated control pipe 192, shown in FIG. 15.

Ex. 1005, 12:17–35 (emphasis added). Petitioner contends the skilled artisan “would have understood that Dukach discloses testing the listen() call’s file descriptor—*i.e.*, its number, or indicator—to determine whether it is on the list of monitored file descriptors—*i.e.*, their numbers, or indicators—to determine whether it is associated with a service to be performed by the front end server.” Pet. 47 (citing Ex. 1002 ¶ 103). Petitioner also contends the skilled artisan “would have also understood that

Dukach’s disclosure of ‘writ[ing] a listen message . . . to [the] file descriptor’s associated control pipe 192’ means an attempt to access the file descriptor’s ‘file,’ in this case the ‘control pipe 192,’ because writing to the control pipe requires access to the pipe.” Pet. 47–48 (citing Ex. 1002 ¶ 103).

Based on Dukach’s intercepted listen() call discussed above, Petitioner contends Dukach discloses “upon an attempt to access a file via a file descriptor, examining at least one stored indicator to determine with what file type the file descriptor is associated,” as recited in claim 1. Pet. 46–48 (citing Ex. 1002 ¶ 103); *see* Pet. 50 (citing Ex. 1002 ¶ 106; Ex. 1005, Fig. 12A).

Dukach also discloses:

When FIG. 11’s back end server calls the OS accept() function in step 160, the call is intercepted by the interposed library’s accept() function 144AC, shown in FIG. 12A. This function is quite similar to the library’s listen() function described above. When it intercepts an accept() call, *step 230 tests if the call’s file descriptor is on the list of monitored file descriptors*. If not, it causes step 238 to pass the call to the OS, terminating the library’s execution for the call.

If the call’s file descriptor has an associated control pipe, indicating it is for a network service handled by the front end, steps 232–236 are executed, starting by having step 232 write an accept message with the call’s parameters to the associated control pipe.

Ex. 1005, 12:57–13:2 (emphasis added). Petitioner again contends the skilled artisan “would have understood that Dukach discloses testing the accept() call’s file descriptor—i.e., its number, or indicator—to determine whether it is on the list of monitored file descriptors—i.e., their numbers, or indicators—to determine whether it is associated with a service to be performed by the front end server—if so, yes; if not, no.” Pet. 48–49 (citing

Ex. 1002 ¶ 104); *see* Pet. 49 (“[The skilled artisan] would have understood that, before the listen() call or the accept() call attempts to access a file using a file descriptor, it examines the file descriptor to determine if it is on the list of monitored file descriptors—the ‘indicators’ of a file type associated with the control pipe for services to be performed by the front end server.” (citing Ex. 1002 ¶ 105; Ex. 1005, Fig. 12A)). Petitioner also contends the skilled artisan “would have also understood that Dukach’s disclosure of ‘write an accept message with the call’s parameters to the associated control pipe’ involves an attempt to access the file descriptor’s ‘file,’ in this case the ‘control pipe 192[,]’ because writing to the control pipe requires access to the pipe.” Pet. 49 (citing Ex. 1002 ¶ 104).

Based on Dukach’s intercepted accept() call discussed above, Petitioner contends Dukach discloses “upon an attempt to access a file via a file descriptor, examining at least one stored indicator to determine with what file type the file descriptor is associated,” as recited in claim 1. Pet. 48–49 (citing Ex. 1002 ¶¶ 104–105); *see* Pet. 50 (citing Ex. 1002 ¶ 106; Ex. 1005, Fig. 12A).

Based on the foregoing evidence, Petitioner contends, and we find, Dukach discloses “upon an attempt to access a file via a file descriptor, examining at least one stored indicator to determine with what file type the file descriptor is associated,” as recited in claim 1. Except as discussed above concerning the limitation “storing at least one indicator of a file type associated with an established file descriptor” (*see* Section II.D.2.a.3, *supra*), Patent Owner does not otherwise contest that this limitation is present in Dukach. *See* generally PO Resp. Any such argument has been waived by Patent Owner. *See* 37 C.F.R. § 42.23(a).

(7) *Conclusion*

For the reasons expressed above, and based on the complete record before us, we conclude that Petitioner has demonstrated by a preponderance of evidence that claim 1 is unpatentable as anticipated by Dukach.

b) *Independent Claims 5, 14, and 17 and Dependent Claims 2 and 7*

Petitioner contends independent claims 5, 14, and 17 and dependent claims 2 and 7 are anticipated by Dukach (Ex. 1005). Pet. 51–63. Patent Owner relies on its same arguments for all of claims 1, 2, 5, 7, 14, and 17, and does not contend that the additional elements in dependent claims 2 and 7 are absent from Dukach. *See generally* PO Resp. 7–17. Nevertheless, for completeness, we address these claims below.

(1) *Dependent Claim 2*

Claim 2 depends from independent claim 1 and further recites “wherein intercepting a system call that creates a copy of at least one file descriptor comprises: intercepting a system call that creates a copy of a file descriptor table,” which Petitioner shows sufficiently to be disclosed by Dukach. *See* Pet. 51–54; Ex. 1001, 14:59–61.

Dukach discloses:

As is well known in the programming arts, a fork creates a child process, which is virtually identical to the parent process which called the fork. *The child process is in a separate process space within the memory space managed by the OS kernel than is the parent process, and has copies of all the variables, data structures, and file descriptors to pipes and network communications which its parent process had.* Forking can be accomplished by calling the OS fork() function, and it usually requires additional steps to be taken by one or both of the two forked processes to prevent them from interfering with the other’s use of pipes and network communications.

Ex. 1005, 9:63–10:7 (emphasis added). Petitioner contends Dukach’s list of monitored file descriptors included in an interposed library discloses a “file descriptor table,” and that the foregoing description discloses “making a fork() call by the parent process (the original) to create a child process (the copy) that is virtually identical to the parent process.” Pet. 51–52 (citing Ex. 1002 ¶ 107).

Dukach also discloses:

The interposed library intercepts fork() call by the back end server processes because such calls not only duplicate the back end server process, but also the copy of the interposed library which is part of each such back end process. Thus, as a result of such a fork there will be a separate interposed library in each of the two resulting back end server processes, each having identical file descriptors to the same control pipes with the front end. Attempts by each such process to communicate over the same control pipe at the same time could result in undesirable problems.

Ex. 1005, 16:54–63; *see id.*, Fig. 19. Petitioner contends the foregoing description discloses that Dukach’s “back end server calls fork() to make a copy of itself, which includes the interposed library, which includes the list of monitored file descriptors.” Pet. 52–53 (citing Ex. 1002 ¶ 107).

Petitioner contends “Dukach discloses that fork() calls create a copy of the list of monitored file descriptors,” which discloses “a system call that creates a copy of a file descriptor table.” Pet. 53–54. Petitioner further contends “Dukach discloses that ‘the interposed library’s fork function 144AF intercepts a back end fork() call,’” which together with the foregoing discloses “intercepting a system call that creates a copy of a file descriptor table.” Pet. 54 (citing Ex. 1002 ¶ 107) (internal citations omitted).

Patent Owner does not contest that this limitation is present in Dukach. *See generally* PO Resp. Any such argument has been waived by Patent Owner. *See* 37 C.F.R. § 42.23(a).

Based on the foregoing evidence, Petitioner contends, and we find, that Dukach discloses “wherein intercepting a system call that creates a copy of at least one file descriptor comprises: intercepting a system call that creates a copy of a file descriptor table,” as recited in claim 2. *See* Pet. 51–54. For the same reasons provided above for independent claim 1, as well as the foregoing arguments and evidence submitted by Petitioner concerning dependent claim 2, we conclude that Petitioner has demonstrated by a preponderance of evidence that claim 2 is unpatentable as anticipated by Dukach.

(2) *Independent Claim 5*

Independent claim 5 recites substantially the same limitations as independent claim 1, except claim 5 specifies the file type associated with a file descriptor is a communication channel, as follows: “intercepting system calls that establish *a communication channel*” and “storing at least one indicator that a file descriptor established by an intercepted system call is associated with *a communication channel*.” Ex. 1001, 15:44–57 (emphases added); *see* Prelim. Resp. 4–5 (Patent Owner acknowledging claim 5 recites limitations similar to claim 1).

Petitioner contends the skilled artisan “would have understood that the bind() system calls disclosed in Dukach establish a communication channel by *defining* a network service socket’s local IP address and port number.” Pet. 54–55 (citing, in part, Ex. 1002 ¶ 110; Ex. 1005, 9:16–37, Fig. 11). Petitioner also contends the skilled artisan “would have understood that

Dukach further discloses that these bind() calls are *intercepted* and that the intercepted bind() calls establish a communication channel, *i.e.*, a control pipe 192.” Pet. 55–56 (citing Ex. 1002 ¶ 110; Ex. 1005, 11:5–26).

Petitioner further contends “Dukach discloses that the interposed library *intercepts* the bind() calls that establish a communication channel, and that the intercepted bind() calls also establish a communication channel (*i.e.*, a control pipe 192) associated with services to be handled by the front end server.” Pet. 56–57 (citing Ex. 1002 ¶ 110).

In addition, Petitioner contends the skilled artisan “would have understood Dukach to teach that the file descriptor established by the dup2() system call, which is part of the *intercepted* bind() call, is associated with the control pipe, *which is a communication channel*, and that the number of the control pipe’s file descriptor (same as that of the bind() call), *i.e.*, its ‘indicator,’ is stored in the list of monitored file descriptors.” Pet. 57–58 (citing Ex. 1002 ¶ 112). Petitioner contends that, “because only file descriptor numbers associated with a control pipe, *i.e.*, a communication channel, are stored in the monitored list of file descriptors, a stored file descriptor number in the monitored list indicates the file descriptor is associated with a communication channel.” Pet. 58 (citing Ex. 1002 ¶ 112). In other words, as testified to by Dr. Greenspun, “a stored file descriptor number, *by its presence in the monitored list*, is an ‘indicator’ that the file descriptor is associated with a communication channel for services to be handled by the front end server.” Pet. 58 (citing, in part, Ex. 1002 ¶ 112; Ex. 1005, 11:27–41) (emphasis added).

Except as discussed above concerning the limitation “storing at least one indicator of a file type associated with an established file descriptor” in

claim 1 (*see* Section II.D.2.a.3, *supra*), Patent Owner does not otherwise contest that Dukach discloses the file type associated with a file descriptor is a communication channel, and provides no additional arguments directed to this claim. *See generally* PO Resp. Any such argument has been waived by Patent Owner. *See* 37 C.F.R. § 42.23(a).

For the same reasons provided above for independent claim 1, as well as the foregoing arguments and evidence submitted by Petitioner concerning claim 5, we conclude that Petitioner has demonstrated by a preponderance of evidence that claim 5 is unpatentable as anticipated by Dukach.

(3) *Claim 7*

Petitioner contends, and we agree, claim 7 recites limitations commensurate in scope with claim 2. Pet. 59. Patent Owner provides no additional arguments directed to this claim. *See generally* PO Resp. For the same reasons provided above for independent claims 1 and 5 and dependent claim 2, we conclude that Petitioner has demonstrated by a preponderance of evidence that claim 7 is unpatentable as anticipated by Dukach. *See* Ex. 1002 ¶ 116.

(4) *Independent Claims 14 and 17*

Petitioner contends independent claims 14 and 17 are substantially the same as independent claims 1 and 5, respectively, except that claims 14 and 17 implement the functions claimed in claims 1 and 5 using program code stored in a computer readable medium, which Dukach plainly discloses. Pet. 59–63 (citing Ex. 1002 ¶¶ 117–131); *see* Ex. 1002 ¶ 124 (Petitioner’s declarant, Dr. Greenspun, testifying that the skilled artisan “would have understood that it was common at the time of (and actually long before) the alleged invention in the ’613 Patent to implement the

functions described in Dukach with program code stored in a computer readable medium.” (citing Ex. 1005, 7:15–20)). We agree with Petitioner that independent claims 14 and 17 recite limitations commensurate in scope with independent claims 1 and 5, respectively (but for the “program code” limitations). Patent Owner provides no additional arguments directed to these claims. *See generally* PO Resp. For the same reasons provided above for independent claims 1 and 5, as well as the foregoing arguments and evidence submitted by Petitioner concerning claims 14 and 17, we conclude that Petitioner has demonstrated by a preponderance of evidence that claims 14 and 17 are unpatentable as anticipated by Dukach. *See* Ex. 1002 ¶¶ 117–131.

E. Obviousness of Claims 1, 2, 5, 7, 14, and 17 over Dukach

Petitioner contends claims 1, 2, 5, 7, 14, and 17 are unpatentable under 35 U.S.C. § 103(a) as obvious over Dukach (Ex. 1005). Pet. 22–63; Pet. Reply 12–24. Petitioner argues this obviousness ground concurrently with its ground that Dukach anticipates these claims, blending and interspersing its arguments in support thereof. *See generally* Pet. 22–63. Although Petitioner has alleged that Dukach discloses every limitation of claim 1, for example, Petitioner also has provided for contingencies in case Dukach were found not to explicitly or inherently disclose certain limitations of claim 1 and instead “teachings” from Dukach were needed to render claim 1 obvious (as opposed to anticipated). *Id.*; *see* Pet. Reply 12–24.

Patent Owner opposes Petitioner’s obviousness contentions here for the same reasons it opposes the anticipation contentions, arguing that an alleged failure of the cited art to disclose or teach certain limitations renders all grounds defective. *See* PO Resp. 7–17. In particular, Patent Owner

argues “while [Petitioner] presents a Ground of obviousness over Dukach, it does not point to any deficiency in Dukach’s disclosure that would be rendered obvious using the level of skill in the art,” and “[Petitioner] merely blankets its ground with conclusory statements that Dukach ‘discloses *or at least renders obvious*’ the various limitations,[] without any distinction between its grounds.” *Id.* at 16. Except as discussed above concerning the limitation “storing at least one indicator of a file type associated with an established file descriptor” in claim 1 (*see* Section II.D.2.a.3, *supra*), Patent Owner does not otherwise contest that Dukach teaches the remaining limitations of claim 1, and provides no additional arguments directed to obviousness of this claim. *See* generally PO Resp. Any such arguments have been waived by Patent Owner. *See* 37 C.F.R. § 42.23(a). Regardless, we disagree with Patent Owner that Petitioner’s obviousness arguments are nothing but conclusory, and find Patent Owner’s arguments here unavailing.

For example, regarding the limitation “storing at least one indicator of a file type associated with an established file descriptor” (*see* Section II.D.2.a.3, *supra*), Petitioner argues, *inter alia*:

The [skilled artisan] would thus have *understood* that Dukach’s “add[ing] the bind() call’s file descriptor [number] to the list of the monitored file descriptors” *results in* “storing at least one indicator of a file type associated with an established file descriptor,” *because* a known way to “add the bind() call’s file descriptor” in the context of computer programming is by “storing” it. Ex. 1002, ¶97. Thus, Dukach renders this limitation obvious.

Pet. 40 (emphases added). That is, Petitioner argues even if Dukach were found not to explicitly or inherently disclose (i.e., anticipate) “storing” of the indicator as recited in this limitation, Dukach would teach or at least fairly suggest (i.e., render obvious) such “storing.” *See* Pet. 40–41 (citing, *inter*

alia, Ex. 1005, 16:15–25 (Dukach disclosing “record[ing]” of “*new_value* in the list of monitored file descriptors.”); Ex. 1002 ¶ 98). Patent Owner offers no evidence to rebut Dr. Greenspun’s testimony in regard to Dukach’s teaching of this limitation, particularly the required “storing.” Indeed, Patent Owner neither cross-examined Dr. Greenspun on any of his testimony nor proffered any rebuttal testimony from its own technical declarant. We credit Dr. Greenspun’s testimony which explains the reasons that one of ordinary skill in the art would have sought to successfully modify Dukach’s “add[ing] the bind() call’s file descriptor [number] to the list of the monitored file descriptors” to include or be performed by “storing” that number in the list (and storing *the list* as a whole). See Ex. 1002 ¶ 97.

The same holds true regarding other claim 1 limitations where Petitioner invoked contingencies of Dukach’s “teachings” in addition to its explicit or inherent disclosures. See, e.g., Pet. 32 (preamble: Dr. Greenspun testifying Dukach teaches the preamble, if limiting, by distinguishing two file types, namely, whether a requested service associated with a file descriptor is to be handled by a front-end server or not (citing, *inter alia*, Ex. 1002 ¶ 90)); Pet. 34 (first “intercepting” limitation: Dr. Greenspun testifying Dukach teaches this limitation because, “as would have been understood by [the skilled artisan], when dup2() outputs (that is, returns) the *new_fd* file descriptor, it creates or produces [i.e., establishes] the *new_fd* file descriptor.” (Ex. 1002 ¶ 92)); Pet. 44–45 (second “storing” limitation: Dr. Greenspun testifying Dukach teaches this limitation “because [the skilled artisan] would have understood that[] ‘record[ing] the alias file descriptor[’s number]’ is basically the same as ‘storing at least one indicator,’ and ‘a duplicate associated with the file descriptor’ is basically

the same as ‘a created copy of a file descriptor.’” (Ex. 1002 ¶ 102 (first alteration added))); Pet. 49–50 (“examining” limitation: Dr. Greenspun testifying Dukach teaches this limitation because certain “pseudo-code of the listen() call and the accept() call provided by Dukach” “teaches testing of the file descriptor to determine if it is associated with a control pipe, meaning the requested service is to be handled by the front end server, which is a file type,” and “[i]f so, the intercepted listen() call or accept() will attempt to access the underlying ‘file,’ in this case the control pipe.” (Ex. 1002 ¶ 106)). Again, Patent Owner offers no evidence to rebut Dr. Greenspun’s testimony in regard to such teachings of Dukach.

Accordingly, for the same reasons provided above for anticipation of independent claim 1 by Dukach (see Section II.D.2.a, *supra*), as well as the foregoing arguments and evidence submitted by Petitioner concerning teachings of Dukach, including the unrebutted testimony of Dr. Greenspun, we conclude that Petitioner has demonstrated by a preponderance of evidence that the teachings of Dukach render claim 1 unpatentable as obvious. See *Cohesive Techs., Inc. v. Waters Corp.*, 543 F.3d 1351, 1364 (Fed. Cir. 2008) (“[I]t is commonly understood that prior art references that anticipate a claim will usually render that claim obvious.”); see also *Game & Tech. Co. v. Activision Blizzard Inc.*, 926 F.3d 1370, 1381 (Fed. Cir. 2019) (“[Appellant’s] assertion that ‘a single reference . . . cannot support obviousness’ is wrong as a matter of law.” (internal citation omitted)); *Arendi S.A.R.L. v. Apple Inc.*, 832 F.3d 1355, 1361 (Fed. Cir. 2016) (“[A] patent can be obvious in light of a single prior art reference if it would have been obvious to modify that reference to arrive at the patented invention.”); *Therasense, Inc. v. Becton, Dickinson & Co.*, 593 F.3d 1325,

1336–37 (Fed. Cir. 2010) (affirming an invalidity judgment where claims were held obvious over a single reference).

Also, for the same reasons provided above for anticipation of claims 2, 5, 7, 14, and 17 by Dukach (*see* Section II.D.2.b, *supra*), as well as the foregoing arguments and evidence submitted by Petitioner concerning teachings of Dukach, including the unrebutted testimony of Dr. Greenspun, we conclude that Petitioner has demonstrated by a preponderance of evidence that the teachings of Dukach render claims 2, 5, 7, 14, and 17 unpatentable as obvious.

F. Obviousness of Claims 1, 2, 5, 7, 14, and 17 over the Combination of Dukach and Dunsmuir

Petitioner contends claims 1, 2, 5, 7, 14, and 17 are unpatentable under 35 U.S.C. § 103(a) as obvious over the combination of Dukach (Ex. 1005) and Dunsmuir (Ex. 1007). Pet. 22–63; Pet. Reply 12–24. Petitioner argues this obviousness ground concurrently with its grounds that these claims are anticipated by or obvious over Dukach, blending and interspersing its arguments in support thereof. *See generally* Pet. 22–63. Although Petitioner has alleged that Dukach discloses or teaches every limitation of claim 1, for example, Petitioner also has provided for contingencies in case Dukach were found not to explicitly or inherently disclose or teach certain limitations of claim 1 and instead teachings from Dunsmuir were needed to render claim 1 obvious (as opposed to anticipated). *Id.*; *see* Pet. Reply 12–24.

Patent Owner opposes Petitioner’s obviousness contentions here for the same reasons it opposes the anticipation contentions, arguing that an alleged failure of the cited art to disclose or teach certain limitations renders

all grounds defective. *See* PO Resp. 7–17. In particular, Patent Owner argues “the duplicative challenges based on Dunsmuir (Ex. 1007) in Ground 3 suffer from the same deficiencies because they still rely on assuming Dukach discloses storing and evaluating an ‘indicator’ of a file type.” *Id.* at 16. Except as discussed above concerning the limitation “storing at least one indicator of a file type associated with an established file descriptor” in claim 1 (*see* Section II.D.2.a.3, *supra*), Patent Owner does not otherwise contest that Dukach or the combination of Dukach and Dunsmuir teaches the remaining limitations of claim 1, and provides no additional arguments directed to obviousness of this claim. *See* generally PO Resp. Any such arguments have been waived by Patent Owner. *See* 37 C.F.R. § 42.23(a). Regardless, we find Patent Owner’s arguments here unavailing.

To support the legal conclusion of obviousness, “there must be some articulated reasoning with some rational underpinning” for combining elements in the manner claimed. *KSR*, 550 U.S. at 418 (quoting *In re Kahn*, 441 F.3d at 988). The test for obviousness is not whether the claimed invention is expressly suggested in any one or all of the references, but whether the claimed subject matter would have been obvious to those of ordinary skill in the art in light of the combined teachings of those references. *See In re Keller*, 642 F.2d 413, 425 (CCPA 1981); *Burckel*, 592 F.2d at 1179 (“[A] reference must be considered not only for what it expressly teaches, but also for what it fairly suggests.”).

Petitioner contends the skilled artisan “would have been motivated to combine Dunsmuir with Dukach because Dukach deals with UNIX system calls and recites the `dup( )`, `dup2( )`, and `fork( )` system calls . . . , which

Dunsmuir expressly teaches.” Pet. 29 (citing Ex. 1005, 11:27–30, 16:15–41; Ex. 1007, 64–65). Petitioner argues the skilled artisan, “when reading Dukach and Dunsmuir together, could and *would* have applied the more detailed descriptions of usage and techniques regarding dup(), dup2(), and fork() calls in Dunsmuir to Dukach.” Pet. 29 (citing Ex. 1002 ¶ 132). “This is because Dukach recites and utilizes dup(), dup2(), and fork() calls and their functionalities, and Dunsmuir provides the explanations on exactly how the dup(), dup2(), and fork() calls operate, thereby further informing [the skilled artisan] how to make and use the system of Dukach expressly relying on dup(), dup2(), and fork() calls.” Pet. 29 (citing Ex. 1002 ¶ 132).

Petitioner argues the skilled artisan “would have had a reasonable expectation of success of combining Dukach and Dunsmuir because both of them deal with UNIX system calls, including the dup(), dup2(), and fork() calls, of which the usage and functionalities are generally standardized and consistent in the art.” Pet. 29–30 (citing Ex. 1002 ¶ 132). Petitioner concludes the skilled artisan “would have reasonably expected to succeed in applying Dunsmuir’s teachings about dup(), dup2(), and fork() calls to Dukach’s dup(), dup2(), and fork() calls because they are substantially identical, and thus, [the skilled artisan] would have readily looked to Dunsmuir to learn more about the well-understood calls described by Dukach.” Pet. 30 (citing Ex. 1002 ¶ 132).

In other words, Petitioner argues, and we agree, even if Dukach were found not to explicitly or inherently disclose or teach certain limitations of claim 1 based on Dukach’s own disclosure of certain UNIX system calls, such as the dup(), dup2(), and fork() system calls, the skilled artisan would have turned to Dunsmuir’s more detailed teachings of usage and techniques

regarding those same system calls, and combined these teachings with those of Dukach to arrive at such limitations. As discussed above, Petitioner's declarant, Dr. Greenspun, offers reasons that the skilled artisan would possess knowledge and motivation to combine the pertinent teachings of Dukach with Dunsmuir, and to do so with a reasonable expectation of success. *See* Ex. 1002 ¶ 132. Patent Owner offers no evidence to rebut Dr. Greenspun's testimony in regard to these combined teachings of Dukach and Dunsmuir, particularly concerning the skilled artisans understanding of `dup( )`, `dup2( )`, and `fork( )` system calls, and we credit Dr. Greenspun's testimony. Based on the foregoing, and on the complete record before us, we are persuaded by Petitioner's cited evidence and explanation of the reasons for combining the teachings of Dukach and Dunsmuir to arrive at the invention of claim 1.

Accordingly, for the same reasons provided above for anticipation of the Challenged Claims by Dukach (*see* Section II.D.2, *supra*) and obviousness of the Challenged Claims over Dukach (*see* Section II.E, *supra*), as well as the foregoing arguments and evidence submitted by Petitioner concerning the combined teachings of Dukach and Dunsmuir, including the unrebutted testimony of Dr. Greenspun, we conclude that Petitioner has demonstrated by a preponderance of evidence that the Challenged Claims are unpatentable as obvious over the combination of Dukach and Dunsmuir.

III. CONCLUSION⁴

Claim(s)	35 U.S.C. §	Reference(s)/ Basis	Claims Shown Unpatentable	Claims Not Shown Unpatentable
1, 2, 5, 7, 14, 17	102(e)	Dukach	1, 2, 5, 7, 14, 17	
1, 2, 5, 7, 14, 17	103(a)	Dukach	1, 2, 5, 7, 14, 17	
1, 2, 5, 7, 14, 17	103(a)	Dukach, Dunsmuir	1, 2, 5, 7, 14, 17	
Overall Outcome			1, 2, 5, 7, 14, 17	

IV. ORDER

Upon consideration of the record, it is:

ORDERED that claims 1, 2, 5, 7, 14, and 17 of U.S. Patent
No. 6,560,613 B1 are unpatentable; and

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

⁴ Should Patent Owner wish to pursue amendment of the challenged claims in a reissue or reexamination proceeding subsequent to the issuance of this 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 its 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).

IPR2021-00361
Patent 6,560,613 B1

For PETITIONER:

Andrew Sommer
GREENBERG TRAURIG LLP
sommera@gtlaw.com

Roshan Mansinghani
Alyssa Holtslander
Jung S. Hahm
UNIFIED PATENTS, LLC
roshan@unifiedpatents.com
alyssa@unifiedpatents.com
jung@unifiedpatents.com

For PATENT OWNER:

Lawrence A. Aaronson
Warren J. Thomas
MEUNIER CARLIN & CURFMAN LLC
laaronson@mcciplaw.com
wthomas@mcciplaw.com