UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

FACEBOOK, INC., EXPEDIA, INC.,
HOMEAWAY.COM, INC., SQUARESPACE, INC.,
WIX.COM, LTD., WIX.COM, INC., GOOGLE LLC,
Petitioners

v.

EXPRESS MOBILE, INC.,
Patent Owner

Case IPR 2021-01455[1]
Patent No. 9,063,755

PATENT OWNER'S SUR-REPLY TO PETITION

[1] Expedia, Inc., HomeAway.com, Inc., SquareSpace, Inc., Wix.com, Ltd., Wix.com, Inc., which filed a petition in IPR2022-00785, and Google LLC, which filed a petition in IPR2022-00790, have been joined as parties to this proceeding.

TABLE OF CONTENTS

TABLE OF AUTHORITIES

Page(s)

COURT DECISIONS

AGENCY DECISIONS

STATUTES

UPDATED TABLE OF EXHIBITS

Exhibit	Description
2001	Declaration of Dr. Kevin C. Almeroth ("Almeroth Decl.")
2002	Curriculum Vitae of Dr. Kevin C. Almeroth
2003	Complaint for Patent Infringement in *Express Mobile, Inc. v. Facebook Inc.*, 6:20-cv-803-ADA (W.D. Tex. 2020)
2004	Complaint for Patent Infringement in *Express Mobile, Inc. v. Google LLC*, 6:20-cv-804-ADA (W.D. Tex.)
2005	Defendants' List of Proposed Claim Terms and Phrases For Construction in *Express Mobile, Inc. v. Facebook Inc.*, 6:20-cv-803-ADA (W.D. Tex.) and *Express Mobile, Inc. v. Google LLC*, 6:20-cv-804 (W.D. Tex.)
2006	Email from Phillip Morton (Tuesday, May 25, 2021, 5:22:52 PM)
2007	Google LLC's Responses to Plaintiff's Second Set of Requests for Production of Documents in *Express Mobile, Inc. v. Google LLC*, 6:20-cv-804-ADA (W.D. Tex.)
2008	Defendant's Invalidity Contentions Regarding U.S. Patent Nos. 6,546,397; 7,594,168; 9,928,044; 9,471,287; and 9,063,755 in *Express Mobile, Inc. v. Facebook Inc.*, 6:20-cv-803-ADA (W.D. Tex.)
2009	Defendants' Responsive Claim Construction Brief in *Express Mobile, Inc. v. Facebook Inc.*, 6:20-cv-803-ADA (W.D. Tex.) [CORRECTED]
2010-2019	*Reserved*
2020	Declaration of Dr. Kevin C. Almeroth ("Second Almeroth Decl.")
2021	June 23, 2020 Claim Construction Order in *Shopify Inc. et al. v. Express Mobile, Inc.*, Case No. 19-439-RGA, (D. Del.)
2022	June 1, 2021 Claim Construction Order in *Express Mobile, Inc. v. GoDaddy.com, LLC*, Case No. 19-1938-RGA (D. Del.)
2023	Joint Claim Construction Statement in *Express Mobile, Inc. v. Facebook, Inc.*, Case No. 6:20-cv-803-ADA (W.D. Tex.)
2024	Defendants' List of Proposed Claim Constructions in *Express Mobile, Inc. v. Facebook, Inc.*, Case No. 6:20-cv-803-ADA (W.D. Tex.)
2025	May 20, 2022, Deposition Transcript of Vijay K. Madisetti, Ph.D.
2026	Board slip opinion, *Ex parte Lee*, Appeal 2016-003036 (PTAB Apr. 3, 2017)
2027	Excerpts from Yakov Fain, The Java Tutorial for the Real World

	(2002)
2028	How do I get Java for Mobile Device?, Java (*available at* https://www.java.com/en/download/help/java_mobile.html)
2029	Defendants' Responsive Claim Construction Brief in *Express Mobile, Inc. v. Facebook, Inc.*, Case No. 6:20-cv-803-ADA (W.D. Tex.)
2030	What is the JDK? Introduction to the Java Development Kit, InfoWorld, (*available at* https://www.infoworld.com/article/3296360/what-is-the-jdk-introduction-to-the-java-development-kit.html)
2031	The Java EE Tutorial
2032	Populating the Page: How Browsers Work, Mozilla, (*available at* https://developer.mozilla.org/en-US/docs/Web/Performance/How_browsers_work)
2033	*Ex parte Orbotech LT Solar, LLC*, Appeal 2011-006601, Application 11/826,336 (May 31, 2012)
2034	Excerpt (Chapter 22) from Nicole-Ambrose-Haynes et al., *Professional ColdFusion 5.0*, (2001)
2035	*Ex parte Evans*, Appeal 2016-000452, Application 12/822,036 (Jan. 23, 2017)
2036	U.S. Patent No. 8,615,216 to Rajguru
2037	January 16, 2014, Office Action Response for U.S. Appl. No. 12/936,395 (U.S. Patent No. 9,063,755)
2038	Mobile Operating Systems' Market Share Worldwide, Statista
2039	Mobile OS Market Share Worldwide, by Month, Zazeinfo (*available at* https://dazeinfo.com/2019/08/23/mobile-os-market-share-worldwide-by-month-graphfarm/)
2040	Deposition of Dr. Vijay Madisetti ("Madisetti Reply Deposition")

I. INTRODUCTION

The first four IPR petitions that argued the '755 patent is obvious were denied institution, and a recent jury trial confirmed the validity of the '755 patent, a finding that was not challenged.[2] This latest attempt should also be rejected.

Now Petitioner relies on a technology called Java Studio Creator (Anderson) to argue obviousness. After Patent Owner exposed Petitioner's improper attempt to cobble together numerous disparate references, Petitioner now asserts that it is "not proposing the Java Studio Creator system that Anderson describes would be modified based on these references." Ex. 1021, ¶ 60.

This admission is fatal because the Anderson reference is far afield from the claimed invention. There is no dispute that the claimed invention comprises a system that produces two codes—an Application and a Player—that are produced by the system's authoring tool to perform discrete functions. Ex. 1001 at 37:26-28 (element 1[b][iv]); 37:43-46 (element 1[c][2]). But Anderson's Java Studio Creator IDE program does not relate to, let alone disclose, producing Player code that can account for different device types. At most, it is concerned with producing a single Java

[2] *See* IPR2021-00709; IPR2021-01144; IPR2021-01228; IPR2021-01471; *Shopify Inc. v. Express Mobile, Inc.*, 19-cv-00439-RGA (D. Del. 2019), Dkt. 421, 3 (verdict); *id.*, Dkt. 437, §§ I-II (declining to challenge validity ruling).

application that can only be executed on devices that include the Java platform.

Because of this fundamental flaw, Petitioner incorrectly asserts that a Java Virtual Machine ("JVM") corresponds to the claimed "Player." But JVM is a known, pre-existing part of the ***Java platform***, and has nothing to do with the code that the ***Java Studio Creator IDE program of Anderson*** generates or produces—or with the claimed Player. Petitioner's reliance on a JVM creates more deficiencies that it purports to solve.

First, a JVM cannot be the claimed Player. Petitioner argues in its Reply that prior claim constructions for Player could include a JVM. Reply at 12-16. But that interpretation of the claim would be contrary to the claim language, the patent's description of this feature of the invention, and the remaining evidence of record. It is clear that the parties dispute that the term can be interpreted to encompass a virtual machine that is part of the platform. Patent Owner submits that the Board should make clear that this limitation should be interpreted to require that a Player is device-specific code which contains instructions for a device and which is separate from the Application, and separate from the operating system, programming language, and platforms of a device. This follows directly from the district court's two constructions—which Petitioner did not dispute—and the intrinsic and extrinsic evidence. *See* II.A, *infra*.

Construing the Player to include a JVM is contrary to the claim language, the

patent's description of this feature of the invention, and all evidence of record. The specification mentions "virtual machine" more than a dozen times and solely treats a virtual machine as a ***platform*** on which the Application and Player can be run—not as Players themselves. Ex. 1001, *e.g.*, 1:63-65 ("Player P . . . extends the operating system and/or virtual machine of the device"); 5:1-5 (referring to "[d]iffering device platforms" as "different operating systems, different versions of an operating system, or different versions of virtual machines on the same operating system"); 7:30-33 ("In one embodiment, the architecture of Player P includes an abstraction interface that separates all device, operating system and virtual machine dependencies from the Player's Application model business logic"); Fig. 2B & 10:63-11:2 (denoting Player as separate entity that runs on top of operating system and virtual machine platforms). The generic use of Java that Petitioner proposes was well-known to be different from the claimed Player.

Dr. Almeroth explained that the Player "facilitates the execution of an Application ***on a machine***," such as a Java Virtual Machine, but it is not the virtual machine itself. Ex. 2020, ¶ 58 (emphasis added). This is disclosed in the specification, which consistently (1) describes the Application and Player as two codes that are executed ***on platforms***, which include virtual machines; and (2) describes the Player and virtual machine as distinct entities. Ex. 2020, ¶¶ 57, 59

(citing Ex. 1001, Fig. 2B, 1:64-67, 6:4-7, 6:48-51, 7:14-20,11:46-52). Petitioner is unable to meaningfully dispute these critical facts which are fatal to the Petition.

Second, there is no evidence that the Java Studio Creator system produces a Player at all—including evidence of any code in the Java Virtual Machine, or what the alleged Player code would be, let alone how that code performs the functionality recited in the claims. Dr. Madisetti could not explain how the Java Studio Creator system (and its authoring tool) generates or produces Java Virtual Machine code as the claims require. Instead, he was reduced to an argument that a Java Virtual Machine is "produced" because it is downloaded onto a computer when Java Studio Creator is installed, or indirectly loaded into memory when a Java application is run. That does not satisfy the claims.

Third, under Dr. Madisetti's faulty theory, there is also no claimed Application, because Petitioner simply asserts there is an Application of Java code, without showing how that code is ***device-independent*** as claimed. Indeed, Petitioner never disputes that its only alleged Application is specific to the Java platform.

Dr. Almeroth's testimony for Patent Owner, moreover, is completely unassailed. Petitioner did not depose Dr. Almeroth. Most of his opinions showing validity are not even contested. Instead of cross-examining Dr. Almeroth, Petitioner filed Dr. Madisetti's Reply declaration, criticizing portions of Dr. Almeroth's

opinions from a safe distance after the close of Patent Owner's expert evidence. Dr. Almeroth's uncrossed, unrefuted testimony should be credited over Dr. Madisetti's.

II. PETITIONER FAILED TO SUBSTANTIATE EITHER GROUND

The POR demonstrated that both grounds fail. The Reply now confirms it.

A. The Reply Fails to Remedy the Numerous Deficiencies Relating to the Claimed Player

1. Petitioner's Combination Does Not Disclose a "Player"

The POR explained that (1) a POSITA would not understand a virtual machine, JVM or otherwise, is the claimed Player, (2) the JVM is not produced by the authoring tool, and (3) the JVM does not contain code that performs the recited functions of the Player. POR, 8-12, 17-20; Ex. 2020, ¶¶ 57-59. The Reply fails to address these deficiencies and instead attempts to sidestep them through a series of untimely, incorrect claim construction arguments. Reply, 12-16.

a) A Java Virtual Machine Cannot Meet the Claimed "Player."

Petitioner asserts that prior claim constructions for the Player encompass a JVM. Reply at 12-16. But that interpretation of the claim is contrary to the claim language, the patent's description of this feature of the invention, and the remaining evidence of record. The intrinsic record consistently treats a virtual machine as part of the device platform alongside an operating system on which the claimed Player executes and is separate from the Player. The claim, therefore, may not be construed to conflate the Player and virtual machines.

Moreover, Petitioner's reliance on these prior constructions for "Player" is misplaced because they were made in a case where no party asserted that a virtual machine could be the claimed Player. These prior constructions did not resolve the issue now before the Board. *See Vivid Techs., Inc. v. Am. Sci. & Eng'g, Inc.*, 200 F.3d 795, 803 (Fed. Cir. 1999).

In prior litigations, a Player was construed to be "device-specific code which contains instructions for a device and which is separate from the Application" and device-dependent code was construed to be "code that is specific to the operating system, programming language, or platform of a device." *See* Ex. 2021, ¶¶ 4-5. A POSITA, in view of the specification, would interpret a Player to be device-specific code which contains instructions for a device and which is separate from the Application, and separate from the operating system, programming language, and platforms of a device.

The specification consistently describes a virtual machine as being in the same category of device environment as an operating system. For example, when describing the device-independent PDL, the specification states that: "[t]he PDL can be conceptually viewed as a device, operating system and **virtual machine agnostic** representation of Java serialized objects." Ex. 1001, 6:49-51. (emphasis added). ***Every single relevant reference in the specification discusses the virtual machine not as a Player, but as a platform of a device.*** *Id.*, 11:46-51 (describing a Player

adapting the Application so that it can be executed by the virtual machine), 1:64-67 (describing a Player extending "the operating system and/or virtual machine"), 7:14-20 (describing a Player running an Application written in PDL can achieve "virtual machine and operating system independence").

Similarly, the file history repeatedly describes the virtual machine as part of a platform of a device equivalent to an operating system in relation to the Player, and emphasized that conventional implementations (like Anderson's) were "written either to the Operating System (OS) or Virtual Machine (VM) of the device," touting the advantage of its new architecture that "extend[ed] the OS and/or VM of the device" using the concept of a "Player" that was introduced by the Patent. (IPR2022-00249, Ex. 1032 at 784-786)[3] ['755 Pros. Hist.]. This intrinsic record also explains that "the player architecture takes full advantage of this by abstracting all the **virtual machine and/or operating system** interfaces from the code," further reinforcing the separation between the Player (the new concept) and the operating system and virtual machines (well known, and on which the Player may run). *Id.* at 515. Indeed, Petitioner's own expert, when asked about Figure 2B—which shows the Player

[3] Petition for *Inter Partes* Review of U.S. Patent No. 9,063,755, Ex. 1032 at 784-786, *Booking Holdings, Inc. v. Express Mobile, Inc.*, Case No. IPR2022-00249 (P.T.A.B. Dec. 1, 2021).

running on a virtual machine/operating system—testified that the virtual machine was not "something different from" the operating system. Ex. 2025, 117:11-117:14.

Petitioner seeks to couch Patent Owner's argument as limited to Figure 2B, but the intrinsic record as a whole makes clear that a virtual machine and a Player are separate and distinct; that this is inherent in the invention's purpose of alleviating dependence on particular operating systems or VMs. *See* Ex. 1001, 1:63-67, 7:14-21. The exemplary discussion of Figure 2B merely reinforces the fact that the patent exclusively describes operating systems and virtual machines as platforms on which Applications and Players run, and that the invention "produce[s] code that" is "executed on the platform." Ex. 1001, 1:40-42. The specification describes "[d]iffering device platforms" as referring to "different operating systems, different versions of an operating system, or different versions of virtual machines on the same operating system." *Id.*, 5:1-5. The Anderson reference itself (Ex. 1003, 35) emphatically describes Java as "a *technology platform*" (emphasis in original). The fact that a Player must run on a platform such as a virtual machine and/or an operating system reinforces that a JVM is not the Player as the construction for Player is properly interpreted.

The Reply fails to reconcile Petitioner's position with the specification's description of the invention. Every reference in the specification to the VM and

Player, including Figure 2B, discloses that the VM and the Player are separate and distinct. Petitioner's extrinsic evidence fails to credibly rewrite that intrinsic record.

Moreover, Dr. Madisetti repeatedly refused to answer questions about Fig. 2B, but now invents a long-winded explanation in his Reply Declaration based on a new and unsupported claim construction. *Compare* Ex. 2025, *e.g.*, 106:22-114:8 (refusing to discuss the disclosure of Fig. 2B, stating "No, I would defer to the language in the specification") *with* Ex. 1021, ¶¶ 30-35. This alone is sufficient reason that Dr. Madisetti's new and belated arguments should be ignored.

The Reply now argues, for the first time, that there is an embodiment of a "native" Player that interacts directly with the operating system without mentioning a virtual machine, and thus that a virtual machine can be a Player. Reply, 15-16. However, the intrinsic record consistently—***exclusively***—treats the Player and virtual machine as separate and distinct concepts. According to the specification, a central purpose of the Player is to solve the problem of "operating system, VM and language fragmentation" and to help the Application achieve "***virtual machine and operating system independence***." *See, e.g.*, Ex. 1001, 1:63-67, 7:14-21. (emphasis added). None of these disclosures would make any sense if the Player can just be the VM, or if the VM was anything more than a platform on which an Application and Player runs. Moreover, Petitioner's argument is illogical: even if a virtual machine were not present in an embodiment, it does not mean that the function and role of

the Player in relation to the rest of the device architecture, including the operating system, has to change and can be replaced by a VM. Instead, in this embodiment, the Application and Player simply execute on a physical machine, rather than a virtual one. No POSITA could read the specification and understand that a JVM (or any virtual machine) can be the claimed Player.

b) The Alleged Authoring Tool Does Not "Produce" a Player.

Even if the JVM could be the claimed Player (it cannot), the authoring tool does not "produce" a Player, as required by the claims. The record shows at most that a "Java Application," as distinct from a JVM, is produced by Anderson's IDE. There is no evidence that Anderson's IDE generates or produces any other set of code.

Petitioner never alleges that ***any*** Player code, let alone a JVM, is produced or generated by Anderson's IDE. Instead, Petitioner argues first that "loading" a JVM into memory and executing code on the JVM is somehow producing a Player, but that cannot be. Claim 1 of the '755 Patent (and claim 1 of related patents such as the '287 patent) requires that the authoring tool produce the Player, and the activation or running of the Player occurs after the Java application is produced. This is confirmed by the recitation of such activation or running in dependent claims such as claim 13 of the '287 patent, which recites, in addition to producing the Player, "The system of claim 1, where said Player is **activated** . . . ." (IPR2021-01456, Ex.

1001 at 39:10-11)[4] ['287 Pat.] (emphasis added); *see also* Ex. 1001 ('755 Pat.), 5:25-28 ("In another alternative embodiment, the Player is activated by a web browser or other software on device 130."). The intrinsic records make clear that activating or executing is a step that ***follows*** production of the Application and Player code by the authoring tool. Thus, merely running a JVM by loading it into memory and executing cannot constitute "producing" a Player.

Petitioner criticizes Patent Owner's argument that the JVM's preexistence on the computer means that its proposed authoring tool cannot "produce" it as required by claim 1, responding that "nothing in claim 1 requires that the Player be obtained from another computer." Reply at 16-17. Petitioner's argument, however, dodges the point. Patent Owner's argument is not that "producing" the Player requires obtaining it from another computer; it is that Petitioner has failed to show that its proposed authoring tool "produces" a Player. Petitioner's argument about what is ***not*** required by the claim does not satisfy its burden of showing what ***is*** required.

Ultimately, the explanation for the lack of a produced Player is simple: the JVM is the platform, a part of the underlying device on which the Java application

[4] Petition for *Inter Partes* Review of U.S. Patent No. 9,471,287, Ex. 1001 at 39:10-11, *Facebook, Inc. v. Express Mobile, Inc.*, Case No. IPR2021-01456 (P.T.A.B. Sept. 1, 2021).

is executed. There is no dispute that a JVM is a preexisting part of the Java platform that exists on a device independently from the Java Studio Creator software. Therefore, most of Petitioner's argument (e.g., that a "Java application cannot even execute without the JVM") (Reply, 16) only further supports Patent Owner's point that a JVM is a part of a device platform and is separate and distinct from the Player.

Petitioner's second theory that storing a copy of the JVM on the computer's storage is "producing" also lacks merit. The claim recites "producing" the Player as separate and distinct from providing the Player to the device. Ex. 1001, 37:29 ("produce a Player"), 37:31-32 ("the Application and Player are provided to the device"). This second theory conflates "producing" and "providing." For example, Petitioner asserts a pre-existing JVM is produced because "dependent claim 11 separately recites that the code for the Player is obtained over a network, and the Petition explained (for Ground 2) that this would have been obvious." Reply, 17. But claim 11 actually contradicts Petitioner's argument. It recites "where said code is **provided** over said network"—which requires that downloading to a device over a network is "providing," ***not*** "producing." Ex. 1001, 38:13-14. (emphasis added).

Moreover, Petitioner's reliance on Ambrose-Haynes' description of installing JDK 1.2 JVM as part of its installation (Pet., 54; Reply, 18) does not substantiate Petitioner's argument, because this example is part of the installation of ColdFusion. Pet., 54 (quoting Ex. 1006, 56). Petitioner does not contend that Anderson

necessarily implements this feature—nor could it. As the Board has acknowledged, "Petitioner is not relying on Ambrose-Haynes to support a modification of Anderson." Inst. Dec., 55. Thus, the fact that Ambrose-Haynes discloses an example of using ColdFusion ***server*** to launch a JVM is irrelevant.

c) The Combination Fails to Render Obvious a "Player" that "Receives the Output Symbolic Name and Corresponding One or More Output Values and Provides Instructions for the Display of a Device to Present an Output Value in the Defined UI Object".

Petitioner's assertion that the JVM "qualifies as a "Player" because it carries out all functions of the Java application" (Reply, 12) exposes another deficiency of the grounds. This argument relies on the "Java application" code as the Player. It fails to identify any distinct code in its alleged "Player" (the JVM) that is a device-dependent code and performs any of the distinct functionality recited in the claims. The Application and Player codes are recited as performing distinct functionalities in claim 1: the Application is software code that fulfills certain requirements (e.g., code that "includ[es] the selected symbolic name of the defined UI object" (element [b][iv])), while the Player is separate software code that fulfills other functional requirements (e.g., "receiv[ing] the output symbolic name and corresponding one or more output values and provides instructions for a display of the device to present an output value in the defined UI object" (element [c][iii])) Reply, 12; Ex. 1001, 37:25-46). Petitioner glosses over this deficiency, and has failed to show that there

is any ***Player*** that is separate from the Application that performs the discrete functionality recited in the claims.

For the Application's functional requirements, Petitioner argued that "[f]or purposes of this claim limitation, the 'Application' corresponds to a Java application produced using the authoring tool." Pet., 47 But for the ***Player,*** Petitioner conceded that it is the Application that contains the code for performing the function of Claim element ***1[c][iii]***—arguing expressly "that ***the Java application*** **"receives the output symbolic name and corresponding one or more output values."** Pet., 69 (underlining and italics emphasis added). Petitioner never identifies a single line of code (by programming language or otherwise) in its alleged Player that performs the separately recited ***Player output*** functionality. This is consistent with the fact that Petitioner has not identified ***any*** prior art code for the Player, except to vaguely reference a JVM.

This glaring flaw is fatal. All Petitioner does to address it is to briefly argue that because "the Java application cannot even function without the JVM, which controls and carries out all of its functions," the JVM is thus also performing the claim step of "receiv[ing] the output symbolic name and corresponding one or more output values." *Id*. But this last-ditch argument fails because the claim requires the Application and the Player to be ***separate*** and to ***each*** contain code for performing their particularized functions. Petitioner's argument would both remove the

separation and distinction between the Application and the Player, ***and*** make the functional limitations recited for the Player meaningless. This is why Petitioner is forced to rely on the production of one code—a Java Application—combined with pure handwaving to argue that the Player limitation is met, allegedly simply because Java Applications run on top of virtual machines.

In fact, as noted above, Petitioner's argument regarding the JVM is just as applicable to the operating system—"the Java application cannot even function without" the operating system because the operating system "carries out the functions of the Java application relating to displaying the user interface"—and thus, by Petitioner's logic, the operating system could also be the claimed Player, contrary to the specification. *See* Pet., 69-70; Reply, 21.

Petitioner's attempt to rely on a device's platform as disclosing a Player should be rejected.

B. Petitioner Fails to Show the Proposed "Application" is a "Device-Independent Code."

Petitioner cannot have it both ways. The claims require an Application that "is a device-independent code." As discussed above, in attempting to argue the existence of the Player, Petitioner is forced to argue that Java Applications (the only code produced by the Anderson IDE) are written for a specific platform—the Java Platform—and are thus not a device-dependent code. But that does not meet the claims.

It also flatly contradicts the specification, which explicitly discloses Java code as an example of device-dependent code:

> Routines 114 may include device-specific routines—that is, codes that are specific to the operating system, programming language, or platform of specific devices 130, and may include, but are not limited to, Java.

Ex. 1001, 3:58-61.[5] Nevertheless, Petitioner maintains that any and all Java applications are device-independent merely by virtue of being specific to the Java platform. *See* Ex. 2040, 86:16-87:17.

Petitioner's primary argument is that the specification describes a single example of Java code being device-independent. Pet., 49-50 (arguing "a Java application" is device-independent because it uses Java, without any analysis of the code within ***the particular*** Java Application being relied on). Even if that reading were correct (it is not), the specification also describes examples of Java code as

[5] Dr. Madisetti attempts to summarily dismiss this disclosure as describing a JVM. Ex. 1021, ¶ 29. This cannot be, because the disclosure refers to Java as "device specific routines" which are "codes," and Dr. Madisetti admitted under cross-examination that a JVM is not Java code. Ex. 2040, 77:24-25. Moreover, the specification describes device-specific routines as Java APIs, not JVMs. Ex. 1001, 10:21-22 ("platform dependent routines 114, such as Java APIs").

device-***dependent***. Moreover, Petitioner's argument that every reference to an "Application," in the specification is necessarily describing device-independent code is incorrect: the specification says that "[t]he Application is ***preferably*** code in a device-independent format," implicitly disclosing that other examples of Java Applications, are device-***dependent***. Ex. 1001, 6:4-5. (emphasis added).

Petitioner does not dispute that the claimed Application is properly construed as "code that is ***not*** specific to the operating system, programming language, or ***platform*** of a device." Reply, 6 (quoting Ex. 2024, 6). Nor does Petitioner dispute that a Java application is run on the Java platform. Indeed, Petitioner maintains that the alleged Application is written in the Java programming language using Java Creator and must be executed by a Java virtual machine. Petitioner's position is that by virtue of being a Java application, the alleged Application must be device-independent. This turns claim construction on its head, effectively arguing that Java code may be device-independent, and the Application must be device-independent, so therefore Java code is the Application.

Casting about for alleged evidence that its Java code is "device-independent" even though it is Java code for the Java platform, the Reply now argues that the phrase "platform of a device" in the agreed construction does not actually mean "platform of a device", but instead really means "hardware and processor" of a device. Reply, 9. Petitioner attempts to substantiate this new position by

distinguishing "machine code" on an "Intel-based microprocessor" from Java bytecode. *Id.*, 7-9. But this attempted distinction is refuted by the specification, which states that "platform[s] of specific devices . . . may include, but are not limited to, Java, Windows Mobile, Brew, Symbian OS, or Open Handset Alliance (OHA)" (*i.e.*, software). Ex. 1001, 3:58-62. The specification makes no mention of any machine code or specific processor, its description of platforms is not limited to a particular "processor" or "hardware," and it states that a device may "include an operating system having a platform that can interpret certain routines." *Id.*, 3:52-55. The specification further provides that "different devices **130** may be operable using different sets of instructions, that is having one of a variety of different 'device platforms.' Differing device platforms may result, for example and without limitation, to different operating systems, different versions of an operating system, or different versions of virtual machines." *Id.*, 4:66-5:5. Thus, the specification provides examples of platforms rooted in software, and even describes a platform as being a virtual machine. As such, "platform" of a device means whether a device has a platform, not a specific "hardware" or "processor." As discussed in the POR at 2-4, many devices at the time of the invention did not, and could not, have the Java platform.

Petitioner tries to change the subject to one of its asserted references, Ambrose-Haynes, which describes Java as platform independent. EX1006, 0055.

But that extrinsic reference's choice of language cannot override the disclosures in the ***specification*** above, which says that Java, for the claimed invention's purposes, is a platform. Platform means platform. Since there is no dispute that the alleged Application depends on the Java platform, it cannot be "device-independent" as construed.

Second, the Reply mischaracterizes the specification's disclosure that "the Application may include Java programming," while ignoring other pertinent disclosures. Reply, 2-3. Tellingly, Petitioner ignores Dr. Almeroth's sworn, uncrossed opinion on this precise point. The intrinsic record explains that while an Application may include internal Java programming, the claimed Application is not a mere Java application. *See* Ex. 1001, 6:55-57 (only "the internal representation of the programming logic is Java"). A key disclosure of the specification is the capability to use the device-independent architecture to extend a Java application, with its internal Java programming logic, to other devices that do not have a specific virtual machine, such as a JVM. *Id.*, 7:30-45 ("the architecture of Player P includes an abstraction interface that separates all device, operating system and virtual machine dependencies"); *Id.*, 7:26-29 ("Compatibility with other languages is inherent based on the various Player abstraction implementations, which may be, for example and without limitation, Java CDC, J2SE or MIDP2 implementations"). In

short, no POSITA could read the intrinsic record and understand the Application term to apply to any and all Java applications.

More importantly, as noted above, even if Petitioner were correct that the specification provides one example of a device-independent Java application (it does not), it also unquestionably provides examples of Java applications that are not. *See, e.g.*, Ex. 1001, 3:58-61. "[T]he Java language is vast and complex," and can be used to create any number of different types and applications with any number of requirements and functionalities. *See* Ex. 1006, 0054. Indeed, Petitioner's Ambrose-Haynes reference demonstrates an example "Application [that] include[s] Java programming" that is unquestionably device-dependent. As Ambrose-Haynes explains, a ColdFusion application can include Java programming. *See, e.g.*, Ex. 1006, 0054. And as the reference explains, an earlier version of ColdFusion was limited to the Windows operating system. *Id.*, 0045.

It is thus clear that applications that include Java programming are not automatically device-independent—and that ***is the only evidence Petitioner provides that the relied-upon Java code is the claimed Application***. *See* Pet., 49-50. Instead, one must look to the particular Java application (bytecode or otherwise) to determine its characteristics—and the specification makes clear that a Java code can be device-dependent, thus failing to meet the claim requirements of the Application. *See, e.g.*, Ex. 1001, 10:21-22 ("…having access to platform dependent

routines 114, such as Java APIs."). Petitioner failed to offer any evidence as to whether the alleged application code could meet the requirements of a device-independent application, or whether, instead, it is in fact device-dependent. Other than broadly arguing that all Java applications are device-independent, Petitioner provides no reason why this specific Java application allegedly produced by ***Anderson*** is device-independent. Petitioner's expert admits that he is not even aware that the Java applications include the ability to detect the browser type, or operating system, or screen sizes. Ex. 2040, 84:4-85:14. Petitioner has thus failed to meet its burden on this point.

II. IT WAS NOT OBVIOUS TO COMBINE THESE DISPARATE REFERENCES.

"A prior art reference must be considered in its entirety, i.e., as a whole, including portions that would lead away from the claimed invention." MPEP § 2141.02 (citing *W.L. Gore & Assoc., Inc. v. Garlock, Inc.*, 721 F.2d 1540 (Fed. Cir. 1983)). A claim cannot be obvious when an omission from a reference "*could* lead away from the claimed invention." Ex. 2026, *Ex parte Lee*, Appeal No. 2016-003036, Application No. 12/822,036, slip op., at 13 (P.T.A.B. Apr. 3, 2017) (emphasis in original). Petitioner does not dispute that its experts never received and never considered vast portions of the alleged references. Nor is there any dispute that a majority of the references are absent from the record. This underscores the

hindsight of Dr. Madisetti's analysis, and that not even a *prima facie* case of obviousness has been made.

Petitioner's only excuse for its expert's failure seems to be to frankly admit that its expert engaged in a hindsight-biased analysis: that is, Petitioner's expert began with the claim limitations and then set out to cherry-pick them out of four voluminous programming manuals, only reading tiny, selected portions thereof. Reply, 22-23. Indeed, as discussed in the POR, Petitioner's expert analysis was so cursory that he did not even fully understand the references. For example, he ignored pertinent disclosures of references because he stopped reading at the heading and did not bother to read the one-paragraph explanation of what the heading meant. POR, 23-24. Whether the references were used for a purportedly "limited purpose" does not vitiate the requirement to fully consider and understand the reference.

The forced, amorphous nature of Petitioner's combination was exposed during Petitioner's expert's latest deposition: when repeatedly asked to describe what is actually included in the proposed combination, Petitioner's expert was unable to do so. *See, e.g.*, Ex. 2040, 52:6-16, 63:1-10, 64:9-17, 70:21-25.73:4-13, 95:16-97:2.

Petitioner seems to suggest it is Patent Owner's fault that pertinent references are not of record for the Board to fully consider. But it is ***Petitioner's*** burden to provide "copies of" its references, 35 U.S.C. § 312 (3)(A). And Petitioner still has

not provided or explained the pertinent disclosures missing from the record. With briefing complete, the record remains incomplete.

A. A POSITA Would Not Have Used Java to Perform Web Services in View of Ambrose-Haynes.

Ambrose-Haynes is a programming manual for ColdFusion. Ex. 1006. Yet Petitioner argues that considering ColdFusion is irrelevant because Petitioner's cherry-picked portions do not rely on ColdFusion. In essence, Petitioner asks the Board to do what Petitioner's expert did: ignore the vast majority of the reference and focus on a portion that is needed to try to pick up claim elements. This is insufficient.

Petitioner's argument that it "is not relying on Ambrose-Haynes for a teaching or motivation to use Java for web services" (Reply, 23) underscores the failure in Petitioner's analysis. "[T]he prior art must be considered as a whole." *Medichem, S.A. v. Rolabo, S.L.*, 437 F.3d 1157, 1166 (Fed. Cir. 2006) (emphasis in original). This art never was. As explained, Ambrose-Haynes includes an entire chapter on web services, which was ***not even considered*** by Petitioner's expert, and which teaches to use, ***not*** Java for web services, but other methods instead. Thus, Petitioner "erred . . . in considering the references in less than their entireties, *i.e.*, in disregarding disclosures in the references that diverge from and teach away from the invention at hand." *W.L. Gore*, 721 F.2d at 1550. While Ambrose-Haynes may praise use of Java in other circumstances, it does ***not*** praise the use of Java for web services,

which is the proposed combination. "Evidence suggesting reasons to combine cannot be viewed in a vacuum apart from evidence suggesting reasons not to combine." *Arctic Cat Inc. v. Bombardier Recreational Prod., Inc.*, 876 F.3d 1350, 1363 (Fed. Cir. 2017).

B. It Would Not Have Been Obvious to Combine the Numerous and Disparate References Teaching Different and Incompatible Platforms.

The Reply also does not explain why a POSITA would have cobbled together many voluminous programming manuals for discrete platforms to make the claimed invention—other than raw hindsight. Instead, ironically, Petitioner claims that Patent Owner "superficial[ly] focus[ed]" on the number of references. Reply, 26. But the Reply does not offer even superficial reasons—other than hindsight—why a POSITA would have found it obvious to look to only these particular snippets of these particular manuals to arrive at the claims.

III. CONCLUSION

The Board should refuse to find that the challenged claims invalid.

Dated: October 27, 2022

Respectfully submitted,

/ Sal Lim /

Sal Lim (Reg. No. 45,706)
Kenneth J. Weatherwax (Reg. No. 54,528)
David L. Alberti (Reg. No. 43,465)
Russell S. Tonkovich (Reg. No. 64,101)
Hong S. Lin (Reg. No. 54,629)
Attorneys for Patent Owner
Express Mobile, Inc.
Attorneys for Patent Owner
Express Mobile, Inc.

CERTIFICATE OF WORD COUNT

Pursuant to 37 C.F.R. § 42.24(d), counsel for Patent Owner Express Mobile, Inc. certifies that this document complies with the type-volume limitation of 37 C.F.R. § 42.24(b). According to Microsoft Office Word's word count, this document contains approximately 5,525 words, including any statement of material facts to be admitted or denied in support, and excluding the table of contents, table of authorities, mandatory notices under § 42.8, exhibit list, certificate of service or word count, or appendix of exhibits or claim listing.

Dated: October 27, 2022

Respectfully submitted,

/ Sal Lim /

Sal Lim (Reg. No. 45,706)
Kenneth J. Weatherwax (Reg. No. 54,528)
David L. Alberti (Reg. No. 43,465)
Russell S. Tonkovich (Reg. No. 64,101)
Hong S. Lin (Reg. No. 54,629)
Attorneys for Patent Owner
Express Mobile, Inc.
Attorneys for Patent Owner
Express Mobile, Inc.

CERTIFICATE OF SERVICE

I hereby certify that, pursuant to 37 C.F.R. § 42.6(e) and with the agreement of counsel for Petitioner, true and correct copies of Patent Owner's Sur-Reply to Petition and Exhibit 2040 are being served electronically on October 27, 2022, to the names and email addresses below:

Heidi L. Keefe	hkeefe@cooley.com
Phil Morton	pmorton@cooley.com
Andrew Mace	amace@cooley.com
Chih Yun (Steve) Wu	swu@cooley.com
Dustin Knight	dknight@cooley.com
Mark R. Weinstein	mweinstein@cooley.com
Adam R. Brausa	abrausa@durietangri.com
Annie Lee	alee@durietangri.com
Troy Smith	tsmith@freeborn.com
David Knapp	dknapp@freeborn.com
Naveen Modi	PH-GoogleExpressMobile-IPR@paulhastings.com
Joseph E. Palys	
Daniel Zeilberger	

Dated: October 27, 2022 By: / Colette Woo /