IN THE UNITED STATES DISTRICT COURT
FOR THE WESTERN DISTRICT OF TEXAS
MIDLAND/ODESSA DIVISION

VIRTAMOVE CORP., Plaintiff, v. GOOGLE LLC, Defendant.	Civil Action No.: 7:24-cv-00033

DEFENDANT GOOGLE LLC'S SUPPLEMENTAL PRELIMINARY INVALIDITY CONTENTIONS[1]

I. INTRODUCTION

Defendant Google LLC ("Google") provides these Preliminary Invalidity Contentions ("Invalidity Contentions") to Plaintiff VirtaMove Corp. ("VirtaMove") for the following patents (collectively, "Asserted Patents" or "Patents-in-Suit") and claims (collectively, "Asserted Claims") identified as asserted in VirtaMove's Preliminary Infringement Contentions ("Infringement Contentions"):

- U.S. Patent No. 7,784,058 ("the '058 patent") – claims 1-5, 10, and 18
- U.S. Patent No. 7,519,814 ("the '814 patent") – claims 1, 2, 4, 6, 9, 10, 13–14[2]

[1] On September 6, 2024, only two business days (and four days total) before the deadline for these contentions, VirtaMove served supplemental infringement contentions. Google has not had time to adequately analyze VirtaMove's supplemental contentions, and accordingly, reserves all rights to subsequently account for and/or address these tardy supplementations.

[2] On June 25, 2024, VirtaMove served its preliminary infringement contentions, which asserted claim 31 of the '814 patent as allegedly infringed. On July 1, 2024, VirtaMove served "corrected" contentions purporting to withdraw its assertions concerning claim 31. Thus, Google does not address Claim 31 herein at this time.

II. RESERVATIONS AND EXPLANATIONS

This disclosure is directed to invalidity under 35 U.S.C. §§ 102, 103, 112 and patent ineligibility under 35 U.S.C. § 101 only and does not address claim construction or non-infringement. *See* Dkt. 34 at 2. Google reserves all rights with respect to such issues, including but not limited to its position that the Asserted Claims are to be construed in a particular manner and are not infringed.

These Invalidity Contentions are preliminary and based on Google's current knowledge, understanding, and belief as to the facts and information available as of the date of these contentions. Google has not yet completed its investigation, discovery, or analysis of information related to this action, and additional discovery may require Google to supplement or amend its Invalidity Contentions. Google reserves the right to amend or supplement their contentions once it gains access to relevant materials VirtaMove has not yet produced. While Google has made a good-faith effort to provide a comprehensive list of prior art relevant to this case, Google reserves the right to modify or further supplement its prior art list and Invalidity Contentions at a later time, including with or based upon pertinent information that may be subsequently discovered from VirtaMove, third parties, or otherwise. Further, Google reserves the right to rely on prior art and invalidity theories disclosed and set forth by defendants in at least the following cases: *VirtaMove, Corp. v. Amazon.com, Inc., Amazon.com Services LLC, and Amazon Web Services, Inc.*, No. 7:24-cv-00030-DC-DTG (W.D. Tex.); *VirtaMove, Corp. v. Hewlett Packard Enterprise Company*, No. 2:24-cv-00093-JRG (E.D. Tex.); and *VirtaMove Corp. v. International Business Machines Corp.*, No. 2:24-cv-00064 (E.D. Tex.). Google also reserves the right to rely on prior art and invalidity theories disclosed in Unified Patents, LLC's Request for *Ex Parte* Reexamination of U.S. Patent 7,784,058, filed September 23, 2024. Moreover, fact discovery has not yet commenced and Google reserves the right to pursue all other defenses that may be available to it, including but not limited

to defenses that the Patents-in-Suit are unenforceable based on laches, estoppel, waiver, acquiescence, patent exhaustion, unfair competition, unclean hands, express or implied license, or any other grounds.

Any invalidity analysis depends, ultimately, upon claim construction, which is a question of law reserved for the Court. Google reserves the right to amend, supplement, or materially modify its Invalidity Contentions in response to any claim construction positions that VirtaMove may take in this case or based on any claim construction the Court may adopt in this case. Google also reserves the right to assert that a claim is indefinite, not enabled, or fails to meet the written description requirement based on any claim construction positions VirtaMove may take in this case or based on any claim construction the Court may adopt in this case.

Google's Invalidity Contentions are directed to the claims asserted by VirtaMove that are actually accused in VirtaMove's Preliminary Infringement Contentions. As Google outlined in its July 22, 2024 letter to VirtaMove, VirtaMove's Preliminary Infringement Contentions are woefully deficient. VirtaMove has not yet remedied the deficiencies identified in that letter. Accordingly, Google reserves the right to modify, amend, supplement or otherwise alter its Invalidity Contentions in the event that VirtaMove supplements or amends its infringement contentions to address these deficiencies or takes a claim construction position that is different than or in addition to those seemingly set forth in its Infringement Contentions.

Further, to the extent VirtaMove's Preliminary Infringement Contentions can be understood at all, VirtaMove appears to be pursuing overly broad constructions of the Asserted Claims in an effort to piece together an infringement claim where none exists and to accuse products that do not practice the claims as properly construed. At the same time and as previously explained with no response by VirtaMove, VirtaMove's Preliminary Infringement Contentions are

in many places too general and vague to discern exactly how VirtaMove contends the Accused Instrumentalities practice each element of the Asserted Claims. *See* July 22, 2024 Letter to VirtaMove re Infringement Contention Deficiencies. Accordingly, these Invalidity Contentions are not intended to be, and are not, an admission that the Asserted Claims are infringed by any of Google's products or technology, that any particular feature or aspect of the Accused Instrumentalities practices any elements of the Asserted Claims, or that any of VirtaMove's proposed constructions are supportable or proper. To the extent that any of the prior art references disclose the same functionality or feature of any of the Accused Instrumentalities, Google reserves the right to argue that said feature or functionality does not practice any element of any of the Asserted Claims, and to argue, in the alternative, that if said feature or functionality is found to practice any element of any of the Asserted Claims, then the prior art reference demonstrates that that element is not novel, is obvious, or is not patentable.

Attached hereto as Exhibits Nos. 814-1 to 814-12, 814-O, 058-1 to 058-16, and 058-O are representative claim charts that demonstrate how the Asserted Claims are invalid in view of certain prior art. The references cited in the attached claim charts may disclose the limitations of the Asserted Claims expressly and/or inherently. Moreover, the suggested obviousness combinations are in the alternative to Google's contentions regarding anticipation. These obviousness combinations should not be construed to suggest that any reference included in any combination is not anticipatory in its own right.

In its Preliminary Infringement Contentions, VirtaMove purports to provide doctrine of equivalents arguments. As Google has set forth in the parties' correspondence regarding VirtaMove's Preliminary Infringement Contentions, those conclusory statements regarding doctrine of equivalents are insufficient to provide notice as to VirtaMove's theories of

infringement. *See* July 22, 2024 Letter to VirtaMove re Infringement Contention Deficiencies. VirtaMove has not taken any action to remedy these deficient contentions. *Id.* Google reserves the right to modify, amend, supplement or otherwise alter its Invalidity Contentions in the event VirtaMove modifies, amends, supplements, or clarifies its Infringement Contentions with respect to direct infringement (literal and under the doctrine of equivalents). Prior art not included in this disclosure, whether known or not known to Google, may become relevant. In particular, Google is currently unaware of the extent to which VirtaMove will contend that limitations of the Asserted Claims are not disclosed in the prior art identified herein. To the extent that such an issue arises, Google reserves the right to identify additional teachings in the same references or in other references that anticipate or would have made the addition of the allegedly missing limitation obvious. Moreover, discovery has not yet begun in this case, but Google may subpoena third parties believed to have information relevant to this disclosure and expressly reserves the right to amend, supplement, or modify this disclosure as information is obtained from third parties, or from VirtaMove itself.

Google further reserves the right to rely on uncited portions of the prior art references and in other publications and testimony as aids in understanding and interpreting the cited portions, as providing context thereto, and as additional evidence that a claim limitation is known or disclosed. Google further reserves the right to rely on uncited portions of the prior art references, other publications, and testimony to establish bases for combinations of certain cited references that render the asserted claims obvious.

The references discussed in the claim charts identified above or elsewhere may disclose the elements of the Asserted Claims explicitly and/or inherently, and/or they may be relied upon to show the state of the art in the relevant time frame. Google further reserves the right to rely on

additional publications, materials, and testimony that are not yet currently identified for purposes other than as prior art, including but not limited to background, state of the art in the relevant time frame, level of ordinary skill in the art, and motivation to combine. The suggested obviousness combinations below are provided in the alternative to Google's anticipation contentions and are not to be construed to suggest that any reference included in the combinations is not by itself anticipatory.

Google is providing Invalidity Contentions only for the claims asserted by VirtaMove, and hereby reserves the right to seek invalidation of all claims in the Patents-in-Suit, including claims 3, 5, 7-8, 11-12, and 15-34 of the '814 patent and claims 6-9, and 11-17 of the '058 patent.

Google reserves the right to modify, amend, or supplement these disclosures as additional information becomes available, and as its discovery and investigation proceed.

III. PRIORITY DATE OF THE ASSERTED PATENTS AND CLAIMS

Google reserves the right to amend its Invalidity Contentions with additional prior art and/or with additional charts of the art if VirtaMove is permitted to change its position and alleges earlier or later priority dates. In particular, in Section E of its infringement contentions cover pleading, which sets out the alleged priority dates for the Asserted Patents, states that the Asserted Claims of the Asserted Patents are entitled to a priority date "***at least as early as***" the filing dates of the earliest provisional applications recited on the face of each of the Asserted Patents. Moreover, VirtaMove states that "[a] diligent search continues for additional responsive information and VirtaMove reserves the right to supplement this response." As stated in Google's July 22, 2024 letter to VirtaMove, VirtaMove's statements regarding priority do not comply with OGP § I, which by its terms requires VirtaMove to identify the "earliest" date of invention, not an "at least as early as" date. And while VirtaMove states a "diligent search continues for additional responsive information," OGP § I requires VirtaMove to have already provided such documents.

Google will object to any attempt by VirtaMove to assert earlier dates than those identified in its PICs or otherwise revise or supplement its contentions contrary to OGP § I. Google also reserves its right to amend its Invalidity Contentions if VirtaMove is permitted to allege earlier dates of conception for the patents-in-suit, or if VirtaMove does not meet its burden to prove the priority dates it asserts.

IV. THE '814 AND '058 PATENTS

1. Identification of Prior Art[3]

Patent References

Google identifies below the patent references presently known to Google that anticipate and/or render obvious the asserted claims of the '814 and '058 patents. Google incorporates by reference all prior art references cited in the patents listed herein and/or their file histories. Google reserves the right to rely upon foreign counterparts of the U.S. Patents identified in these Invalidity Contentions, U.S. counterparts of foreign patents and foreign patent applications identified in these Invalidity Contentions, U.S. and foreign patents and patent applications corresponding to articles and publications identified in these Invalidity Contentions, and any systems, products, or prior inventions related to any of the references identified in these Invalidity Contentions. To the extent the following patents disclose and describe particular products and/or software programs that were publicly known and/or in public use prior to the priority date of the '814 and '058 patents, in addition to each patents itself serving as a prior art reference under 35 U.S.C. § 102, Google may rely on the various products and/or software programs described in the patents as grounds for invalidity under 35 U.S.C. § 102 because they were in public use, in which case it would have

[3] To the extent one or more prior art patents or publications are identified in the claim charts attached to this document but are not included in the tables and lists for the '814 or '058 Patents, those prior art patents or publications should also be considered as prior art to the '814 and '058 Patents, respectively.

been obvious to a person of ordinary skill in the art to combine the actual systems in public use with the patents describing those systems because the patents described the systems in public use and refer to them throughout. In some of these cases, Google may serve subpoenas and/or otherwise request information pertaining to the products and/or software programs (and/or obtain the actual products and/or software programs themselves) once fact discovery begins. To the extent necessary, Google will amend and/or supplement these Invalidity Contentions based on information received in response. The following patents are prior art under at least 35 U.S.C. §§ 102(a), (b), (e), (f), (g), and/or 103.

'814 Patent:

Country of Origin	Patent Number	Publication Date	Priority Date
United States	2003/0014466 ("Berger")	January 16, 2003	June 29, 2001
United States	2003/0233490 ("Blaser")	December 18, 2003	June 12, 2002
United States	2002/0095479 ("Schmidt 479")	July 18, 2002	January 18, 2001
United States	2002/0066022 ("Calder")	May 30, 2002	November 29, 2000
United States	2002/0120660 ("Hay")	August 29, 2002	February 28, 2001
United States	2002/0133529 ("Schmidt 529")	September 19, 2002	January 16, 2001
United States	2002/0138629 ("Schmidt 629")	September 26, 2002	March 22, 2001
United States	2002/0124072 ("Tormasov")	September 5, 2002	February 16, 2001
United States	6,529,985 ("Deianov")	March 4, 2003	February 4, 2000
United States	7,437,556 ("Tucker")	October 14, 2008	May 9, 2003

'058 Patent:

Country of Origin	Patent Number	Publication Date	Priority Date
United States	5,375,241 ("Walsh")	December 20, 1994	December 21, 1992
United States	5,574,915 ("Lemon")	November 12, 1996	December 21, 1993
United States	5,696,970 ("Sandage")	December 9, 1997	April 1, 1993
United States	5,708,811 ("Arendt")	January 13, 1998	February 26, 1993

United States	5,845,118 (“Gheith”)	December 1, 1998	December 14, 1995
United States	6,173,336 (“Stoeckl”)	January 9, 2001	August 31, 1994
United States	6,212,574 (“O’Rourke”)	April 3, 2001	April 4, 1997
United States	6,349,355 (“Draves”)	February 19, 2002	February 6, 1997
United States	2002/0052727 (“Bond”)	May 2, 2002	October 30, 2000
United States	2002/0091867 (“Reid”)	July 11, 2002	January 8, 2001
United States	2002/0095479 (“Schmidt ’479”)	July 18, 2002	January 18, 2001
United States	2002/0120660 (“Hay”)	August 29, 2002	February 28, 2001
United States	2002/0124072 (“Tormasov”)	September 5, 2002	February 16, 2001
United States	6,453,460 (“Keyes”)	September 17, 2002	April 26, 1999
United States	2002/0133529 (“Schmidt ’529”)	September 19, 2002	January 16, 2001
United States	2002/0138629 (“Schmidt ’629”)	September 26, 2002	March 22, 2001
United States	6,529,985 (“Deianov”)	March 4, 2003	February 4, 2000
United States	2003/0065856 (“Kagan ’856”)	April 3, 2003	October 3, 2001
United States	6,567,974 (“Czajkowski ’974”)	May 20, 2003	February 25, 2000
United States	2003/0103455 (“Pinto ’455”)	June 5, 2003	November 30, 2001
United States	2003/0177285 (“Hunt”)	September 18, 2003	June 16, 1999
United States	2003/0233490 (“Blaser”)	December 18, 2003	June 12, 2002
United States	WIPO Pub. No. 2004/001615 (“Andjelic”)	December 31, 2003	June 19, 2002
United States	2004/0003137 (“Callender”)	January 1, 2004	June 26, 2002
United States	2004/0025165 (“Desoli ’165”)	February 5, 2004	August 5, 2002
United States	2004/0022256 (“Green ’256”)	February 5, 2004	July 30, 2002
United States	6,698,015 (“Moberg ’015”)	February 24, 2004	June 13, 2000

United States	2004/0064644 (“Lin ’644”)	April 1, 2004	September 30, 2002
United States	2004/0078600 (“Nilsen ’600”)	April 22, 2004	July 11, 2002
United States	2004/0103417 (“Voellm”)	May 27, 2004	November 27, 2002
United States	2004/0111549 (“Connor ’549”)	June 10, 2004	December 10, 2002
United States	2004/0139171 (“Chen ’171”)	July 14, 2004	November 25, 2002
United States	2004/0142563 (“Fontarensky ’563”)	July 22, 2004	January 16, 2003
United States	2004/0190558 (“Oliver ’558”)	September 30, 2004	March 31, 2003
United States	2004/0249933 (“Govindarajan ’933”)	December 9, 2004	June 4, 2003
United States	2005/0038827 (“Hooks ’827”)	February 17, 2005	August 11, 2003
United States	6,976,037 (“D’Souza”)	December 13, 2005	March 27, 2000
United States	6,988,271 (“Hunt ’271”)	January 17, 2006	October 2, 1998
United States	7,080,172 (“Schmalz ’172”)	July 18, 2006	May 27, 2003
United States	7,213,247 (“Wilner ’247”)	May 1, 2007	January 10, 2000
United States	7,424,710 (“Nelson ’710”)	September 9, 2008	December 18, 2002
United States	7,437,556 (“Tucker”)	October 14, 2008	May 9, 2003

Publications

Google identifies below the publications presently known to Google that anticipate and/or render obvious the asserted claims of the ’814 and ’058 patents. Google incorporates by reference all prior art references cited in the publications listed herein. To the extent the following publications disclose and describe particular products and/or software programs that were publicly known and/or in public use prior to the priority date of the ’814 and ’058 patents, in addition to each publication itself serving as a prior art reference under 35 U.S.C. § 102, Google may rely on the various products and/or software programs described in the publications as grounds for invalidity under 35 U.S.C. § 102 because they were in public use, in which case it would have been obvious to a person of ordinary skill in the art to combine the actual systems in public use with the published documents describing those systems because the documents described the systems in public use and refer to them throughout. In some of these cases, Google may serve subpoenas and/or otherwise request information pertaining to the products and/or software programs (and/or obtain the actual products and/or software programs themselves) once fact discovery begins. To the extent necessary, Google will amend and/or supplement these Invalidity Contentions based on information received in response. The following publications are prior art under at least 35 U.S.C. §§ 102(a), (b), (e), (f), (g), and/or 103.

’814 Patent:

Title	Date of Publication	Author(s)	Publisher
Jails: Confining the omnipotent root. (“Kamp”)	May 22, 2000	Poul-Henning Kamp, Robert N. M. Watson	The FreeBSD Project

Freeing your Computer from the Hardware ("Sapuntzakis")	March 23, 2002	Constantine P. Sapuntzakis, Ramesh Chandra, James C. Norris, Monica S. Lam, Mendel Rosenblum	Computer Science Department – Stanford University
Supporting Ubiquitous Computing with Stateless Consoles and Computation Caches ("Schmidt Paper")	August 6, 2000	Brian Keith Schmidt	The Computer Science Department and the Committee on Graduate Studies of Stanford University
Thinstall Studio Help ("Thinstall Handbook")	On or before October 17, 2003	Jonathan Clark	Thinstall
Virtual private servers and security contexts ("Vserver Guide")	On or before August 4, 2003	Jacques Gélinas	Solucorp
Using jails in FreeBSD for fun and profit ("Hope")	June 2002	Paco Hope	Usenix & Sage: The Advanced Computing Systems Association & The System Administrators Guild

'058 Patent:

Title	Date of Publication	Author(s)	Publisher
Writing a Dynamic-Link Library for Windows ("MASM 6.1 Chapter 10")	1992	Mike Eddy	Microsoft
Programming Windows 95 ("Petzold 1996")	1996	Charles Petzold	Microsoft
The Windows NT Kernel Architecture ("Solomon 1998")	October 1998	D.A. Solomon	IEEE
FIFS: A Framework for Implementing User Mode File Systems in Windows ("Almeida 1999")	June 1999	D. Almeida	USENIX

Unifying the User and Kernel Environment ("Draves 1997")	March 1997	R. Draves	Microsoft
Supporting Ubiquitous Computing with Stateless Consoles and Computation Caches ("Schmidt Paper")	August 6, 2000	Brian Keith Schmidt	The Computer Science Department and the Committee on Graduate Studies of Stanford University
Jails: Confining the omnipotent root. ("Kamp")	May 22, 2000	Poul-Henning Kamp, Robert N. M. Watson	The FreeBSD Project
Freeing your Computer from the Hardware ("Sapuntzakis")	March 23, 2002	Constantine P. Sapuntzakis, Ramesh Chandra, James C. Norris, Monica S. Lam, Mendel Rosenblum	Computer Science Department – Stanford University
Thinstall Studio Help ("Thinstall Handbook")	On or before October 17, 2003	Jonathan Clark	Thinstall
Virtual private servers and security contexts ("Vserver Guide")	On or before August 4, 2003	Jacques Gélinas	Solucorp
Using jails in FreeBSD for fun and profit ("Hope")	June 2002	Paco Hope	Usenix & Sage: The Advanced Computing Systems Association & The System Administrators Guild

Prior Art Systems

Google identifies below the systems or software products presently known to Google that anticipate and/or render obvious the asserted claims of the '814 and '058 patents. Google incorporates by reference all prior art references cited in the publicly available materials listed herein. Although Google's investigation continues, information available to date indicates that

each system or software product was (i) known or used in this country before the alleged invention of the claimed subject matter of the asserted claims; (ii) was in public use and/or on sale in this country and/or was the subject of a printed publication more than one year before the filing date of the patent; and/or (iii) was invented by another who did not abandon, suppress, or conceal, before the alleged invention of the claimed subject matter of the asserted claims. While Google is producing publicly available materials uncovered to date regarding these systems, its investigation regarding the functionality embodied in these systems is ongoing. In some of these cases, Google may serve subpoenas and/or otherwise request information pertaining to the products and/or software programs (and/or obtain the actual products and/or software programs themselves) once fact discovery begins, and expects to supplement its contentions as additional information is provided, for example through third-party subpoenas and further discovery from VirtaMove. The following systems or software products, as well as any systems associated with or reflected in any of the patents or publications cited herein, are prior art under at least 35 U.S.C. §§ 102(a), (b), (f), (g), and/or 103.

’814 Patent:

System/Service	Date Became Public	Entities Involved in Public Use/Sale
FreeBSD	November 1993	Berkeley Software Distribution
Solaris	May 2002	Sun Microsystems
Thinstall	October 2003	VMware
vServer	May 26, 2003	Linux
Virtuozzo	January 2002	SWsoft

’058 Patent:

System/Service	Date Became Public	Entities Involved in Public Use/Sale
FreeBSD	November 1993	Berkeley Software Distribution
Solaris	May 2002	Sun Microsystems

Thinstall	October 2003	VMware
vServer	May 26, 2003	Linux
Virtuozzo	January 2002	SWsoft

1. Whether Each Item of Prior Art Anticipates or Renders Obvious the Asserted Claims of the ’814 and ’058 Patents

Each of the Asserted Claims is invalid because it fails to meet one or more requirements for patentability. The individual bases for invalidity, including whether and how each item of prior art anticipates each Asserted Claim or renders it obvious, are provided in the charts attached as Exhibit Nos. 814-1 to 814-12, 814-O, 058-1 to 058-16, and 058-O. Each of the foregoing listed prior art references, the underlying work, and/or the underlying apparatus or method qualifies as prior art under one or more subsections of 35 U.S.C. § 102 or § 103.

Although Google has identified at least one citation per limitation for each reference, each and every disclosure of the same limitation in the same reference is not necessarily identified. Rather, in an effort to focus the issues, Google has generally cited representative portions of identified references, even where a reference may contain additional support for a particular claim element. In addition, persons of ordinary skill in the art generally read a prior art reference as a whole and in the context of other publications and literature. Thus, to understand and interpret any specific statement or disclosure within a prior art reference, such persons would rely on other information within the reference, along with other publications and their general knowledge. Google may rely upon uncited portions of the prior art references and on other publications and expert testimony to provide context, and as aids to understanding and interpreting the portions that are cited. Google may also rely on the prior art of record for any permissible purpose, including prior art discussed in the Patents-in-Suit themselves, including to show that the Patents-in-Suit are anticipated or obvious, show the state of the art, show motivation to combine a reference with one

or more other references, and to show the proper scope of the claims. Google may also rely on uncited portions of the prior art references, other disclosed publications, and the testimony of experts to establish that a person of ordinary skill in the art would have been motivated to modify or combine certain of the cited references so as to render the claims obvious.

Anticipation

Some or all of the Asserted Claims are invalid as anticipated under 35 U.S.C. § 102 under the prior art references identified above and in the claim charts included in Exhibit Nos. 814-1 to 814-12, and 058-1 to 058-16. which identify specific examples of where each limitation of the Asserted Claims is found in the prior art references. As explained above, the cited portions of prior art references identified in the attached claim charts are exemplary only and representative of the content and teaching of the prior art references, and should be understood in the context of the reference as a whole and as they would be understood by a person of ordinary skill in the art.

Obviousness

Each anticipatory prior art reference, either alone or in combination with other prior art, also renders the asserted claims obvious to one of ordinary skill in the art. To the extent any limitation is deemed not to be exactly met, either explicitly or inherently, by an item of prior art listed above and in Exhibit Nos. 814-1 to 814-12 and 058-1 to 058-16, then any purported differences are such that the claimed subject matter as a whole would have been obvious to one skilled in the art at the time of the alleged invention, in view of the state of the art and knowledge of those skilled in the art, as shown in Exhibit Nos. 814-O and 058-O. In particular, each anticipatory prior art reference on its own renders obvious the claimed inventions and also may be combined with (i) information known to persons skilled in the art at the time of the alleged

invention and/or (ii) any of the other anticipatory prior art references. The item of prior art would, therefore, render the relevant claims invalid for obviousness under 35 U.S.C. § 103(a).

The Supreme Court clarified the standard for what types of inventions are patentable. *See KSR Int'l Co. v. Teleflex, Inc.,* 127 S. Ct. 1727, 1731 (2007) ("[T]he combination of familiar elements according to known methods is likely to be obvious when it does no more than yield predictable results."). In particular, the Supreme Court emphasized that inventions arising from ordinary innovation, ordinary skill or common sense should not be patentable. *See id.* at 1732, 1738-39, 1742-43, 1746. Because the asserted patents simply combine elements well known in the art in a straightforward fashion to achieve a well-known and obvious result and thus yield no more than one skilled in the art would expect from such combinations, the claims of those patents are obvious. The asserted claims are therefore invalid under 35 U.S.C. § 103 because they do nothing more than combine known techniques and apparatuses according to their known and ordinary uses to yield predictable results.

The Supreme Court further held that "[w]hen a work is available in one field of endeavor, design incentives and other market forces can prompt variations of it, either in the same field or a different one. If a person of ordinary skill can implement a predictable variation, § 103 likely bars its patentability. For the same reason, if a technique has been used to improve one device, and a person of ordinary skill in the art would recognize that it would improve similar devices in the same way, using the technique is obvious unless its actual application is beyond his or her skill ...." *Id.* at 1740. Accordingly, a person of skill in the art would have been motivated to combine or adapt known or familiar methods in the art, especially where market forces prompt such variations.

In view of *KSR*, the United States Patent and Trademark Office issued a set of new Examination Guidelines. See Examination Guidelines for Determining Obviousness Under 35

U.S.C. § 103, 72 Fed. Reg. 57526 (Oct. 10, 2007). These Guidelines identify various rationales under *KSR* for finding a claim obvious at the time of the filing of the application for this patent, including those based on other precedents, including but not limited to:

(A) Combining prior art elements according to known methods to yield predictable results;

(B) Simple substitution of one known element for another to obtain predictable results;

(C) Use of known techniques to improve similar devices (methods, or products) in the same way;

(D) Applying a known technique to a known device (method, or product) ready for improvement to yield predictable results;

(E) "Obvious to try" – choosing from a finite number of identified, predictable solutions, with a reasonable expectation of success;

(F) Known work in one field of endeavor may prompt variations of it for use in either the same field or a different one based on design incentives or other market forces if the variations would have been predictable to one of ordinary skill in the art;

(G) Some teaching, suggestion, or motivation in the prior art that would have led one of ordinary skill to modify the prior art reference or to combine prior art reference teachings to arrive at the claimed invention. See 72 Fed. Reg. 57529.

These rationales apply in rendering obvious the asserted claims of the asserted patents. Here, the state of the art at the relevant time expressly taught and suggested using application virtualization to allow for multiple application/operating system images to co-exist on a single compute platform. Further, the state of the art at the relevant time expressly taught and suggested utilizing shared libraries to allow computing systems to efficiently run multiple software applications without conflicts or performance issues. As a result, one skilled in the art would have

known to combine or modify references that described known systems and methods which one of skill in the art would have recognized as offering improvements to solutions at that time. Each of the prior art references identified herein described systems and methods that were known to offer such improvements, and, accordingly, one of skill in the art would have been motivated to combine or modify the references as described herein.

In light of the various reasons to combine discussed in detail below, invalidity under 35 U.S.C. § 103 can be based on one or more references disclosed herein. If and to the extent VirtaMove argues that one or more of these references, combined with the knowledge of persons of ordinary skill in the art at the relevant time, fails to disclose one or more specific elements of an asserted claim, Google reserves the right to rely on the combination of one or more of the references disclosed herein, with one or more other references disclosed herein. Further, to the extent VirtaMove argues that claim elements that are not disclosed in the specifications of the asserted patents and/or related applications were enabled, Google reserves the right to rely on the combination of one or more of the references disclosed herein with one or more other references disclosed herein.

Reasons exist to combine one or more of the references included in these Invalidity Contentions with each other. Generally, reasons to combine any of these references with others exist within the references themselves, as well as within the knowledge of those of ordinary skill in the art at the relevant time. For example, many of these references identify and address the same issues and suggest very similar solutions to those issues in the field of systems for application virtualization and resource sharing in multi-application environments. If and to the extent that VirtaMove challenges the correspondence of any of these references with respect to particular elements of the Asserted Claims, Google reserves the right to supplement these Invalidity

Contentions to identify additional motivations to combine particular references with one another to the extent necessary. Google may rely upon a subset of the references or all of the references depending upon further investigation. Google's contentions that the references in this section, in various combinations, render the Asserted Claims obvious under 35 U.S.C. § 103 are in no way an admission or suggestion that each reference does not independently anticipate the Asserted Claims under 35 U.S.C. § 102. Any of the references disclosed herein, may be combined with other references disclosed herein, to render obvious, and therefore invalid, each of the Asserted Claims.

With respect to the prior art references disclosed in the attached exhibits, a person of ordinary skill in the art would have been motivated to combine any of them for a number of reasons, including the reasons set forth in the attached exhibits and/or as described herein.

'814 Patent

As the '814 patent concedes, virtual machines were well-known in the art at the time of the claimed invention. '814 patent at 1:51-2:3. Virtual machines "offer[] the ability for multiple application/operating system images to effectively co-exist on a single compute platform." *Id*. The supposed "limitations" of such virtual machines were the requirement that "an operating system, including files and a kernel, must be deployed for each application" as opposed to "one operating system regardless of the number of application containers deployed." *Id.* According to VirtaMove, the "invention" of the '814 patent "taught a secure container system 'to allow applications to more effectively share a common compute platform, and also allow applications to be easily moved between platforms, without the requirement for a separate and distinct operating system for each application.'" Dkt. 42 at 1-2. However, the '814 patent also concedes that application virtualization, which does not require "a separate operating system" for each

application, was well-known in the art at the time of the claimed invention. *Id.* at 2:4-12. The prior art application virtualization solutions therefore, admittedly, solved the so-called "limitations" of virtual machines.

The '814 patent also asserts that existing application virtualization solutions did not "isolate applications into distinct environments" such that the applications "possess a unique identity." *Id.* According to VirtaMove, the "invention" of the '814 patent provides a "novel and unconventional solution" in which "applications are associated with 'secure containers,' where each application has its own copies of files and a unique identity, but is allowed to 'contend for common resources and utilize different versions of system files.'" Dkt. 42 at 1-2. But as the '814 patent concedes, the noted prior art application virtualization solutions "provide[] a degree of separation of an application from the underlying operating system" (*id.*), and as such, it would have been obvious to a person of skill in the art to similarly separate individual applications such that they are "isolated" within "distinct environments" and thus possess a distinct identity.

A person of ordinary skill in the art would have been further motivated to combine the prior art references cited herein to achieve such a solution.

The prior art references are directed to, for example, isolating applications into distinct environments and/or virtualization. *See, e.g.*, Calder at [0002] ("The invention relates to distributed computing, and more particularly, relates to secure peer-to-peer Internet or enterprise distributed computing. The invention also relates to the secure execution of an application on a client computer."); Schmidt 479 at [0002] ("The present invention relates to virtual namespaces for active computing environments."); Hay at [0003] ("The present invention relates to operating systems for computers. More specifically, the present invention relates to a method and an apparatus for associating virtual servers identifiers with processes within an operating system,

wherein the operating system supports multiple virtual servers on a single computing platform."); Tormasov at [0003] ("This invention relates to the provision of full independent computer system services across a network of remote computer connections."); Schmidt 529 at [0002] ("The present invention relates to the representation and encapsulation of active computing environments."); Schmidt 629 at [0002] ("The present invention relates primarily to the field of computer networks, and in particular to migrating open network connections."); Schaefer at [0002] ("The present invention relates to computer software, and more particularly to operating system software."); Berger at Abstract ("A system and method are disclosed which enable management of compartments implemented by an OS for defining containment in a system."); Blaser at [0012] ("The inventions relate generally to computer systems having facilities for providing virtual portions of file systems and configuration settings to applications. More particularly, the inventions relate to computer systems that provide a layer organization for files and configuration settings that can be overlaid on top of an operating system."); Kamp at 1 ("The FreeBSD 'Jail' facility provides the ability to partition the operating system environment, while maintaining the simplicity of the UNIX 'root' model."); Sapuntzakis at 1 ("The new model features the decoupling of software execution from the computer hardware by encapsulating all software execution in an abstraction we call *capsules*."); Hope at 48 ("The kernel mediates access to global system information and network resources, controls the creation and use of special devices, and logically isolates jailed processes from the main system and from each other."); Schmidt Paper at v ("Capsules provide a private, portable, persistent, customizable computing environment with active processes, i.e. they are self-contained, running computations."). Therefore, a person of ordinary skill in the art looking to execute applications in distinct environments and/or virtualize applications would have been motivated to consult and combine features from these references.

A person of ordinary skill in the art would also have been motivated to combine these references to address common issues related to and/or virtualization, such as the lack of "isolate[d] applications." *See, e.g.*, Calder at [0076] ("The interception module acts as a "virtual layer" between the operating system and the application" which can "prevent the application program from accessing certain files and directories on the client machine."); Schmidt 479 at [0018] ("This enables the capsule to migrate between binary compatible machines and provides privacy and isolation between multiple capsules."); Hay at [0036] ("Also note that each virtual environment 120 and 121 appears to be operating on a separate dedicated computer system, whereas in reality, virtual environments 120 and 121 operate on the same computer system."); Tormasov at [0026] ("If two processes in different virtual computing environments were started for execution from one file (for example from the shared file system) they would be completely isolated from each other, but use the same set of read-only shared physical memory pages."); Schmidt 529 at [0038] ("Processes in the same capsule may communicate with each other and share data via standard IPC mechanisms, for instance using pipes, shared memory, or signals. Communication with processes outside the capsule, on the other hand, is restricted to Internet sockets and globally shared files."); Schmidt 629 at [0035] ("Processes in the same capsule may communicate with each other and share data via standard Interprocess Communication (IPC) mechanisms, for instance using pipes, shared memory, or signals. Communication with processes outside the capsule, on the other hand, is restricted to Internet sockets and globally shared files."); Schaefer at [0019] ("Referring to FIG. 2, two separate applications 52,54, or two instances of the same application (50 illustrated in FIG. 1), can be provided private contexts in which they will appear to have separate or differing copies of system services, configuration and data."); Berger at [0035] ("Services and processes (e.g., applications) on the system may be run within separate compartments. Processes within each

compartment may only have direct access to the resources in that compartment. Access to other resources, whether local or remote, may be allowed only via well-controlled communication interfaces."); Blaser at [0025] ("Provided in one aspect of the invention are application layers which are isolated from other applications on a computer."); Kamp at 3 ("In doing so, we simultaneously maintain the existing UNIX security model, allowing multiple users and a privileged root user in each jail, while limiting the scope of root's activities to his jail."); Sapuntzakis at 1 ("By running multiple capsules on the same machine, we can isolate the computations such that they are protected from each other, they can be given different access rights, and they do not interfere with each other."); Hope at 49 ("The kernel mediates access to global system information and network resources, controls the creation and use of special devices, and logically isolates jailed processes from the main system and from each other."); Schmidt Paper at 33 ("Compute capsules embody a user's personal computing environment. As such, we would like them to be isolated from each other, capable of being customized, and managed by the underlying system.").

A person of ordinary skill in the art would have known that isolating applications in this way is beneficial for multiple reasons, such as increasing security and preventing use collisions. *See, e.g.*, Calder at [0005] ("By allowing a distributed process to execute on the consumer's machine, the task may, among other things: (i) cause a system malfunction; (ii) improperly access confidential information; or (iii) otherwise adversely affect the performance of their computer."); Schmidt 479 at [0014] ("Other benefits from machine-centric computing that are desirable to retain are privacy, isolation, and security. In particular, machine centric computing gives the user a dedicated machine that is completely secure and private, and totally isolated from everyone else."); Schaefer at [0003] ("Typically, in certain prior art systems, great pains were taken to modify a

client system to appear as if a program was installed, or to actually install the software itself and then back out these modifications to restore the original configuration. In doing this, mile problems present themselves: conflicts between an application and the computer's current configuration, multiple instances of the same or different applications, complexity of the back out process requires an application to be put through a rigorous process to ensure all of its modifications can be accounted for, and the use of shared files and system components by multiple applications complicates back out and the installation process.”).

A person of ordinary skill in the art would also have been motivated to combine these references to reduce the need for a separate “operating system, including files and a kernel,” for each application. *See, e.g.*, Calder at [0139] (“For these files, all low level file manipulation APIs are passed through the interception module in the virtual layer 415. Instead of calling the local operating system kernel to perform the file operation, the operation is communicated over the network 130 to another computer or the server 120.”); Schmidt 479 at [0039] (“In one embodiment, the operating system is re-partitioned so that some of the internal program's state is moved into the capsule. This includes moving one or more elements of the CPU state, the file system state, the device state, the virtual memory state, and the inter-process communication (IPC) state into the capsule. ”); Hay at [0036] (“Also note that each virtual environment 120 and 121 appears to be operating on a separate dedicated computer system, whereas in reality, virtual environments 120 and 121 operate on the same computer system.”); Tormasov at [0025] (“However, all of the virtual computing environments share the same kernel of the operating system. All the processes inside the virtual computing environment are the common processes of the operating system and all the resources inherent to each virtual computing environment are shared in the same way as typically happens inside an ordinary single kernel operating system.”); Schmidt 529 at [0039] (“To provide

such functionality, the traditional operating system is re-partitioned as shown in FIG. 1 so that all host-dependant and personalized elements of the computing environment are moved into the capsule 100, while leveraging policies and management of the shared underlying system 105."); Schmidt 629 at [0037] ("To provide such functionality, the traditional operating system is re-partitioned as shown in FIG. 1 so that all host-dependant and personalized elements of the computing environment are moved into the capsule 100, while leveraging policies and management of the shared underlying system 105."); Schaefer at [0032] ("Many components used by operating systems and running applications are shared across several applications or instances."); Berger at [0039] ("According to one exemplary OS in which various embodiments of the present invention may be implemented, containment functionality is achieved by means of kernel-level mandatory protection of processes, files and network resources."); Blaser at [0033] ("Shared libraries (such as DLLs), system accessible configuration (such as registry entries), and version control are managed by the layering subsystem, optionally using an internal database."); Kamp at 6 ("On a box making use of the jail facility, we refer to two types of environment: the host environment, and the jail environment. The host environment is the real operating system environment, which is used to configure interfaces, and start up the jails. There are then one or more jail environments, effectively virtual FreeBSD machines."); Sapuntzakis at 1 ("Capsules can contain executing software as well as all the state needed to support the execution. This can include multiple processes, files, and other software including libraries or even a complete operating system."); Hope at 49 ("The kernel mediates access to global system information and network resources, controls the creation and use of special devices, and logically isolates jailed processes from the main system and from each other."); Schmidt Paper at 36-37 ("Our approach is to add a new interface to the operating system so that capsules can import and export critical kernel state,

i.e. the operating system merely caches the state. Although the kernel manages all state information, ownership has been re-partitioned between capsules and the operating system. Compute capsules own user-specific and host-dependent information, while the underlying system maintains global state (see Figure 2-1).”).

A person of ordinary skill in the art would have known that reducing the need for multiple operating systems in this way is beneficial for multiple reasons, such as reducing resource costs and network resources. *See, e.g.*, Tormasov at [0013] (“One solution to these problems is the use of computer emulators. The OS/390 operating system for IBM mainframe computers has been in use for many years (Samson). The same products with hardware partitioning are produced by another vendor of computers—Sun Microelectronics (Kobert). Each personal computer user is given a fully-functional virtual computer with emulated hardware. This approach is very costly because the operating system installed in the corresponding virtual computer does not recognize the existence of the neighboring analogous computers and shares practically no resources with those computers. Experience has shown that the price associated with virtual computers is very great.”); Sapuntzakis at 8 (“The term capsule was introduced earlier by one of the authors and Brian Schmidt[14]. In this work, capsules were implemented at the operating system. Hardware resources were virtualized by modifying the operating system kernel. The advantage is that the capsules are much lighter weight than virtual-machine based capsules.”); Hope at 49 (“The primary disadvantage is that simulating the hardware can be expensive in terms of system resources. Each instance of the OS must have its own disk resources for its file system. The hardware virtualization and mediation is not free; it takes CPU cycles away from the applications themselves. In some contexts a completely separate instance of the operating system is overkill for the level of isolation that is needed.”).

<u>’058 patent</u>

In the ’058 patent specification, it purports to alleviate the problems to the functionality of applications “caused by conflicts for shared resources.” ’058 patent at 1:27. According to VirtaMove, allowing two applications to share the same critical system element poses security risks: for example, if one application requires the use of a range of network ports, a second application sharing the same networking element would be exposed to the security risk of similarly handling that range of ports. Dkt. No. 42 at 3 (citing ’058 patent at 5:54-6:3. According to VirtaMove, the ’058 Patent instead teaches replicating certain critical system elements into the context of individual software applications using shared libraries. *Id.* (citing ’058 patent at 5:21-25). It discloses providing an “additional service in the form of a CSE,” where “[s]hared libraries are used as a mechanism whereby an application can utilize a CSE that is part of a library.” *Id.* (quoting ’058 patent at 5:35-41). According to VirtaMove, the patent explains that “each application has its own unique data space. This indivisible data space ensures that CSEs are unique to an application or more commonly to a set of applications associated with a container[.]” *Id.* (quoting ’058 patent at 3:30-45). In this design, the CSEs are “replicated, and embodied in the context of an application,” in contrast to prior art systems. *Id.* (quoting ’058 patent at 7:22-8:3).

As the ’058 patent concedes, however, computing systems capable of running multiple software applications that share resources were well known in the art at the time of the claimed invention. *Id.* at 1:21-24 (“Computer systems are designed in such a way that software application programs share common resources. It is traditionally the task of an operating system to provide mechanisms to safely and effectively control access to shared resources.”); *see also* FIGS 1, 2a, and 2b (each showing prior art system architectures where multiple applications share resources). The ’058 patent asserts that “the centralized control of elements, critical to software applications,

hereafter called critical system elements (CSEs) create[d] a limitation caused by conflicts for shared resources" and thus the concept of executing CSEs in the context of an application is an "important distinction between [its] invention and prior art systems and architectures." *Id.* at 1:25-28. The '058 patent also discloses that a "common solution" to conflicts between shared resources was "to place software applications that may potentially conflict on separate compute platforms" and that the state of the art at the time the '058 patent was filed "define[d] two architectural approaches to the ***migration of critical system elements from an operating system into an application context***." *Id.* at 1:33-37 (emphasis added). The '058 patent goes on to state that in "existing systems and architectures, regardless of where a service is defined to exist, that is, in kernel mode, in user mode as a single process or in user mode as multiple processes, all support the concept of a single shared service." *Id.* at 1:50-54.

Even if that were true, however, given that these prior art systems already implemented the concept of executing CSEs in the context of an application, it would have been obvious to a person of ordinary skill in the art to remedy the known problems associated with a single shared service—conflicts and performance/security issues—in the same way the '058 patent did: by utilizing shared libraries as a way of duplicating critical system elements provided by the operating system. *Id.* at 49-51; *see also id.* at 5:24-31; 7:63-8:15; '058 Prosecution History, Feb. 18, 2009 Amendment at 8 ("the claimed invention utilizes such shared libraries as a way of duplicating critical system elements provided by the operating system.").

A person of ordinary skill in the art would have been further motivated to combine the prior art references cited herein to achieve such a solution. The prior art references are directed to, for example, resource sharing in multi-application environments. *See, e.g.*, Walsh at Abstract ("A dynamic-link library method and system for providing services to one or more application

programs."); Sandage at Abstract ("The Card Services DLL [dynamic linked library] maintains a Card Services database of the Card Services resources available such that the Card Services DLL can share the Card Services resources among clients of Card Services. Windows applications can call the functions in the Card Services DLL directly to access Card Services functions."); Reid at [005] (solving "a need for better ways to enable application programs to communicate and Share data in an object-oriented fashion."); Keys at Abstract ("A computer System according to various aspects of the present invention includes an environment having a single processing space, i.e. not designed for multiprocessing with a process switch. In such an environment, multiple application programs may refer to common library program specifications without conflict."); Czajkowski at Abstract (" The applications may utilize or share one or more 'original' classes."); Hunt at Abstract ("Operating system functions are defined as objects that are collections of data and methods. The objects represent operating system resources. The resource objects can be instantiated and used across process and machine boundaries."); D'Souza at Abstract ("A version of a shared component is stored in a local directory with an application that uses that particular version. Another version of the shared component exists on the system and is useable by any number of other computer programs."); Andjelic at 1:14-17 ("Important operating system functions include sharing hardware among users, preventing users from interfering with each other, resource scheduling, organizing data for secure and rapid access, and supporting I/O functions and network communications."); Voellm at [0020] ("In the first method, processing routines for manipulating the data are associated with initial dynamic link library (i.e., initial.dll) 204. It is appreciated that initial.dll 204 may actually be more than one dynamic link library that is associated with app.exe 202."); Almeida at 3 ("FIFS allows the developer to write a simple user-mode dynamic link library (DLL), using traditional operating system facilities, programming libraries, and development tools. . . . It has

the potential to perform all file system operations available to NT user-mode programs, including file locking, byte range locking, and directory notification."); Draves at 1 ("We advocate structuring operating systems to unify the user and kernel environments. The operating system should present a single environment, with common interfaces or APIs, common run-time characteristics for scheduling, paging, stack usage, and shared libraries, common debugger and development tools, etc."); Solomon at 44 ("Under Windows NT, user applications do not call the native Windows NT OS services directly; rather, they go through one or more environment subsystem dynamic-link libraries (DLLs)."). Therefore, a person of ordinary skill in the art looking to design an architecture or system to efficiently manage resources shared by multiple applications would have been motivated to consult and combine features from these references.

A person of ordinary skill in the art would also have been motivated to combine these references to reduce common issues related to sharing resources between applications, such as conflicts between the different needs of different applications. *See, e.g.*, Walsh 2:28-36 ("Since a dynamic-link library is linked at run time, not with a linker, a copy of the library is not inserted into the application program's executable file. Instead, a copy of the library is loaded into memory while the application program is running. As a result, the application program and dynamic-library are always physically distinct. Such distinctness allows the application program and dynamic-link library to be updated, compiled, and tested separately from each other."); Draves at 7 ("In any case, the kernel should have its own copy of the shared library's global data in the same way that each user process loading the shared library gets its own copy of global data."); Reid at [0023] ("Thus, referring to FIG. 2, the share object 19 creates the class object 12 which is shared by the applications 10a and 10b. The shared memory 17 is used by both applications 10a and 10b. Thus, each application 10a and 10b may have member data in the class object 12 and this data is specific

to one application's address space."); Keyes at 8:42-46 ("In a method according to various aspects of the present invention, multiple application programs that would otherwise behave incorrectly due to non-independent references to shared data may be executed in a single processing environment without conflict."); Czajkowski at 5:28-37 ("The general approach taken by the system and method as disclosed herein is to create new classes to replace the original classes in order to isolate the execution of the applications, such that different applications cannot typically access the same static fields. In one embodiment, only one copy of each original class is maintained, regardless of how many applications utilize it. Classes are transparently and automatically modified, so that each application has a separate copy of its static fields."); Hunt at [0013] ("Different versions of the same resource have different identifiers. This ensures that applications that need a specific version of a resource can receive that version.").

A person of ordinary skill in the art would also have been motivated to combine these references to alleviate the performance degradation associated with switching process modes. *See, e.g.*, Callender [0020] ("Having corresponding operations implemented in both user mode implementation 130 and kernel mode implementation 140 is desirable for at least a couple of reasons. . . . Given the overhead of switching from kernel mode to user mode (and switching back again), kernel mode implementation 140 calling user mode implementation 130 (for operations that can be implemented in user mode) typically would result in less than optimal performance . . . ."); Solomon at 1 ("Under Windows NT, user applications do not call the native Windows NT OS services directly; rather, they go through one or more environment subsystem dynamic-link libraries (DLLs)."); Sandage 3:25-32 ("A Card Services dynamic linked library (DLL) contains a library of C functions which perform the Card Services functions. The Card Services DLL maintains a Card Services database of the Card Services resources available such

that the Card Services DLL can share the Card Services resources among clients of Card Services. Windows applications can call the functions in the Card Services DLL directly to access Card Services functions."); D'Souza at 3 at 10-15 ("Applications that employ DLL/COM redirection use the versions of any shared components that are installed in the application directory, regardless of what versions are installed elsewhere on the System. To isolate an application, the application is Stored in a directory with the shared component that is being isolated."); Almeida at 3 ("By running in usermode, FIFS addresses the issues of portability and ease of programming. Rather than forcing the developer to write to the more volatile kernel-mode programming environment, FIFS allows the developer to write a simple user-mode dynamic link library (DLL), using traditional operating system facilities, programming libraries, and development tools. . . . It has the potential to perform all file system operations available to NT user-mode programs, including file locking, byte range locking, and directory notification."); Andjelic at Fig. 1 (teaching an architecture that involves a user-space driver enabling direct communication between an application and a network interface card); Voellm at [0005] ("In one embodiment, new functionality is effectively added to the computer application by launching the computer application in a suspended mode. An asynchronous procedure call (APC) is registered with user process associated with the computer application. When the user process is unsuspended, the APC loads an additional dynamic link library (DLL) to be associated with the computer application. The additional DLL includes routines that operated differently than routines originally associated with the computer application. The references to the routines within the computer application are redirected to the routines of the additional DLL.").

A person of ordinary skill in the art would have known that by utilizing shared libraries as a way of duplicating critical system elements provided by the operating system, they could remedy

the known problems associated with a single shared service—conflicts and performance/security issues. *See, e.g.*, D'Souza at 3 at 10-15 ("Applications that employ DLL/COM redirection use the versions of any shared components that are installed in the application directory, regardless of what versions are installed elsewhere on the System. To isolate an application, the application is Stored in a directory with the shared component that is being isolated."); Callender at [0025] ("InfiniBand application 210A first calls user mode InfiniBand library 220A which knows about user mode library 230A via specified configuration information held in a registry database. In one Sense, user mode library 230 may be thought of as an extension of user mode InfiniBand library 220A. The user mode library 230A exports a well know application program interface ("API") (in the form of a dispatch table) for the routines it Supports. In one embodiment, user mode library 230A is implemented as a dynamic link library ("DLL). Although not necessarily required, much of the benefits of InfiniBand are realized through the kernel-bypass IO features offered by the user mode library 230A."); Walsh 2:28-36 ("Since a dynamic-link library is linked at run time, not with a linker, a copy of the library is not inserted into the application program's executable file. Instead, a copy of the library is loaded into memory while the application program is running. As a result, the application program and dynamic-library are always physically distinct. Such distinctness allows the application program and dynamic-link library to be updated, compiled, and tested separately from each other."); Draves at 7 ("In any case, the kernel should have its own copy of the shared library's global data in the same way that each user process loading the shared library gets its own copy of global data.");

One particular way of doing this, as claimed in the '058 patent, is to make critical system elements stored in the shared library functional replicas of the OS critical system elements, which when called by the application is run in the context of the application. This solution was also

disclosed in the prior art. *See, e.g.*, D'Souza at 6:14-20 ("For example, to activate DLL/COM redirection for an application called 'my app.exe,' an empty file named 'myapp.exe.local' is created and stored in the same directory where myapp.eXe is installed, e.g., a pointer referencing myapp.exe.local is Stored in the same directory as a pointer referencing myapp.exe."); *id.* at 7:6-18 ("As a more Specific example, consider an application named "c:\myapp\myapp.exe' that calls a library loading routine ("LoadLibrary" in WINDOWS operating systems) using the path name "c:\programfiles\common files\System\mydll.dll." The directory ("c:\myapp") where the application resides is searched for a file named "myapp.exe.local." If that file is found, then the directory ("c:\myapp') is searched for an isolated, or local, version of "mydll.dll". If such a file is found in the directory, the library loading routine will load "c:\myapp\mydll.dll." Otherwise, the library loading routine will load the global version of the DLL/COM "c:\Windows\System\mydll.dll."); Callender at [0022] ("In accordance with the present invention, however, and as described in more detail with respect to the example hardware driver model illustrated in FIGS. 2A-2C, user mode implementation 130 and kernel mode implementation 140 share a common interface. (In FIG. 1, this common interface is part of user mode implementation 130 and kernel mode implementation 140 and is not shown separately.) Accordingly, the same source code may be written for a kernel mode communication operation and a user mode communication operation. Using the same source code (e.g., a static library derived from a common source) is a significant benefit that reduces development time, testing time, and programming errors because less program code needs to be written and tested."); *id.* at [0025] ("InfiniBand application 210A first calls user mode InfiniBand library 220A which knows about user mode library 230A via specified configuration information held in a registry database. In one sense, user mode library 230 may be thought of as an extension of user mode InfiniBand library

220A. The user mode library 230A exports a well-known application program interface ('API') (in the form of a dispatch table) for the routines it supports. In one embodiment, user mode library 230A is implemented as a dynamic link library ('DLL')."); *id.* at [0033] The process mode independent library (e.g., vendor specific library 232C and vendor specific library 242B) are statically linked into both miniport 240C and user mode library 230C. As noted above, by making the library a common component, the development, maintenance, and testing of an HCA miniport may be reduced. Generally, the PMI library's functions have one or more context specific extension areas passed to them. The extensions are PMI work areas, used by the PMI function to establish the processing context. Based on the processing context, the corresponding user mode or kernel mode implementations are called or mapped for a commonly named function.").

Google may rely on cited or uncited portions of the prior art, other documents, and expert testimony to establish that a person of ordinary skill in the art would have been motivated to modify or combine the prior art so as to render the claims invalid as obvious. In the event VirtaMove challenges the motivation to combine particular prior art references, Google reserves the right to supplement these contentions to further specify the motivation to combine the prior art to the extent necessary.

2. Charts Identifying Specifically Where and How in Each Prior Art Each Asserted Claim is Found

The claim charts attached as Exhibit Nos. 814-1 to 814-12, 814-O, 058-1 to 058-16, and 058-O specify where the limitations of the Asserted Claims are disclosed in the prior art. This identification is based on Google's current understanding of VirtaMove's apparent interpretation of the Asserted Claims as reflected in its infringement contentions. These Invalidity Contentions shall not be construed as an admission of or acquiescence to VirtaMove's proffered or apparent claim constructions and interpretations or of other positions advanced by VirtaMove during the

course of this litigation. Furthermore, they shall not prevent Google from advancing its claim construction and/or noninfringement positions.

These Invalidity Contentions are also intended to be exemplary in nature rather than exhaustive. Google has endeavored to identify the most relevant portions of the prior art references. The references, however, may contain additional support for a particular claim limitation. Google may rely on uncited portions of the prior art references, other documents, fact witness testimony, VirtaMove's testimony, and expert testimony to provide context or to aid in understanding the cited portions of the references. Where Google cites to a particular figure in a reference, the citation should be understood to encompass the caption, related description of the figure, and any text relating to the figure. Conversely, where Google cites to particular text referring to a figure, the citation should be understood to include the figure as well. Google reserves the right to rely on any portion of each reference in this chart, cited or not, to show that the asserted claims are invalid.

'814 Patent:

Exhibit No.	Exhibit (charts explaining bases for invalidity of '814 patent)[4]
814-1	Berger
814-2	Blaser
814-3	Schmidt '479
814-4	Kamp
814-5	Sapuntzakis
814-6	Schmidt Paper
814-7	FreeBSD
814-8	Solaris
814-9	Thinstall and Thinstall Handbook
814-10	Vserver and Vserver Guide
814-11	Virtuozzo
814-12	Tucker

[4] An identification of a prior art reference for purposes of identifying an exhibit should be understood to mean that the chart discloses not only bases for invalidity based on that reference, but also that reference in combination with other references identified or referred to in the chart.

814-O	Calder, Hay, Hope, Schmidt ’529, Schmidt ’629, Schaefer, Tormasov, and Deianov

’058 Patent:

Exhibit No.	Exhibit (charts explaining bases for invalidity of ’058 patent)
058-1	Almeida 1999
058-2	Bond ’727
058-3	Callender ’137
058-4	Draves 1997
058-5	D’Souza ’037
058-6	Kamp
058-7	Sandage ’970
058-8	Sapuntzakis
058-9	Schmidt ’479
058-10	Schmidt Paper
058-11	FreeBSD
058-12	Solaris
058-13	Thinstall and Thinstall Handbook
058-14	VServer and VServer Guide
058-15	Virtuozzo
058-16	Tucker
058-O	Reid ’867, Walsh ’241, Hunt ’285, Czajkowski ’974, Blaser, Keyes ’460, Andjelic ’479, Solomon 1998, Voellm ’417, Lemon ’915, Arendt ’811, Fontarensky ’563, Gheith ’118, Hooks ’827, Hunt ’271, Moberg ’015, Nelson ’710, Wilner ’247, Kagan ’856, Desoli ’165, Nilsen ’600, Oliver ’558, O'Rourke ’574, Pinto ’455, Green ’256, MASM 6.1 Chapter 10, Schmalz ’172, Chen ’171, Draves ’355, Stoeckl ’336, Connor ’549, Govindarajan ’93, Lin ’644, Hay, Hope, Schmidt ’529, Schmidt ’629, Tormasov, and Deianov

Google’s attached charts and tables provide citations to certain prior art references where information that anticipates and/or renders obvious the claimed inventions may be found, either expressly or inherently. There may, however, be additional support in the cited references or other grounds for Google’s contentions that such prior art meets a particular claim element, and Google reserves the right to supplement these Invalidity Contentions with such information. For example,

persons of ordinary skill in the art at the time of the filing of the Patents-in-Suit knew to read prior art references as a whole (as opposed to excerpts in isolation), and in the context of other publications and literature as well as the general knowledge in the field. Google may rely on all such information, including uncited portions of the prior art references listed herein, and on other publications and testimony, to provide context and as aids to understanding and interpreting the listed references, or to establish that a person of ordinary skill in the art would have been motivated to modify or combine any of the cited references so as to render the claims obvious.

Some of the references on which Google relies are prior art systems that were known, sold, and/or used prior to the priority date of the Patents-in-Suit. Google is continuing to collect documentation, software, and source code regarding these prior art systems, including through third party discovery. Google reserves the right to supplement its contentions to include any further information obtained in discovery.

V. IDENTIFICATION OF LIMITATIONS WHICH ARE INDEFINITE UNDER 35 U.S.C. § 112(B) OR LACK WRITTEN DESCRIPTION UNDER 35 U.S.C. § 112(A)

1. Indefiniteness

Google identifies the following additional grounds for invalidity of the Asserted Claims based on 35 U.S.C. § 112 ¶ 2. Google notes that its investigation is continuing, and fact discovery has not yet begun. Google therefore is identifying those issues under 35 U.S.C. § 112 ¶ 2 of which it is currently aware. Google reserves the right to supplement and/or amend its contentions if VirtaMove changes its infringement contentions or allegations in any way.

Google's contentions regarding 35 U.S.C. § 112 ¶ 2 shall not be construed as an admission that any claim construction advanced by Google in this case is in any way inconsistent, flawed, or erroneous. Nor should these contentions prevent Google from advancing any claim construction and/or noninfringement positions. Further, these contentions shall not be construed as an

admission of or acquiescence to VirtaMove's apparent claim construction and interpretations or of other positions advanced by VirtaMove during the course of this litigation.

Google contends that one or more of the Asserted Claims are invalid as indefinite under 35 U.S.C. § 112 ¶ 2. 35 U.S.C. § 112, ¶ 2 requires that a patent "conclude with one or more claims particularly pointing out and distinctly claiming the subject matter which the applicant regards as his invention." 35 U.S.C. § 112, ¶ 2. This "definiteness standard" requires "'clear notice of what is claimed, thereby appris[ing] the public of what is still open to them.'" *Interval Licensing LLC v. AOL, Inc.*, 766 F.3d 1364, 1370 (Fed. Cir. 2014) (quoting *Nautilus, Inc. v. Biosig Instruments, Inc.*, 134 S. Ct. 2120, 2128–29 (2014)). The "statute requires . . . 'that a patent's claims, viewed in light of the specification and prosecution history, inform those skilled in the art about the scope of the invention with reasonable certainty.'" *Id.* (quoting *Nautilus*, 134 S. Ct. at 2129).[5]

The following claim limitations of the Asserted Claims are indefinite and therefore invalid:

- **'814 Patent:** "disparate computing environment" (claim 1), "secure, executable, applications related to a service" (claim 1), "a unique root file system that is different from an operating system's root file system" (claim 1), "an operating system's" (claim 1), "including a kernel a set of associated local system files" (claim 1), "the applications each include an object executable by at least some of the different operating systems" (claim 1), "residing permanently" (claim 1), "remain resident" (claim 1), "compatible with a local

[5] Google has also provided invalidity contentions under 35 U.S.C. § 102 and § 103 under VirtaMove's apparent claim constructions, all of which seem to contend that the claims are definite. Google's provision of § 102 and § 103 arguments are in no case an admission that the Asserted Claims are definite.

kernel" (claim 1), "copies or modified copies of the associated local system files that remain resident on the server" (claim 1), "execution file" (claim 2).

- **'058 Patent:** "critical system elements" (claim 1), "OS critical system elements" (claim 1), "shared library critical system elements" (claim 1), "functional replicas of OSCSEs" (claim 1), "some of the SLCSEs stored in the shared library . . . are accessible to some of the plurality of software applications" (claim 1), "forms a part of the one or more of the plurality of software applications" (claim 1), "an instance of a SLCSE provided to at least a first of the plurality of software applications from the shared library is run in a context of said at least first of the plurality of software applications" (claim 1), "a unique instance of a corresponding critical system element" (claim 1), "a SLCSE related to a predetermined function is provided to the first of the plurality of software applications" (claim 1); "a SLCSE for performing a same function is provided to the second of the plurality of software applications for running a second instance of the SLCSE simultaneously" (claim 1), "unique access to a unique instance of a CSE" (claim 10); "SLCSEs are not copies of OSCSEs" (claim 18).

2. Lack of Written Description

Google identifies the following additional grounds for invalidity of the Asserted Claims based on 35 U.S.C. § 112 ¶ 1. Google notes that its investigation is continuing and discovery is ongoing. Google therefore is identifying those issues under 35 U.S.C. § 112 ¶ 1 of which it is currently aware. Google reserves the right to supplement and/or amend its contentions if VirtaMove changes its infringement contentions or allegations in any way.

Google's contentions regarding 35 U.S.C. § 112 ¶ 1 shall not be construed as an admission that any claim construction advanced by Google in this case is in any way inconsistent, flawed, or erroneous. Nor should these contentions prevent Google from advancing any claim construction and/or noninfringement positions. Further, these contentions shall not be construed as an admission of or acquiescence to VirtaMove's apparent claim construction and interpretations or of other positions advanced by VirtaMove during the course of this litigation.

Google contends that one or more of the Asserted Claims are invalid for failure to comply with the written description requirements under 35 U.S.C. § 112 ¶ 1. The written description, drawings, and claims in a patent must clearly allow a person of ordinary skill in the art to understand and recognize that the patentee invented what is claimed. *Gentry Gallery, Inc. v. Berkline Corp.*, 134 F.3d 1473, 1479 (Fed. Cir. 1998). In this regard, the patent must demonstrate by disclosure in the specification to those skilled in the art that the patentee had "possession" of what is now asserted to be the claimed invention. *Vas-Cath Inc. v. Mahurkar*, 935 F.2d 1555, 1561 (Fed. Cir. 1991). The written description must actually or inherently disclose every claim element. *PowerOasis, Inc. v. T-Mobile USA, Inc.*, 522 F.3d 1299, 1306-07 (Fed. Cir. 2008). It is not enough to say that undisclosed subject matter would have been obvious or within the normal skill set of a person of ordinary skill. *ICU Medical, Inc. v. Alaris Medical Sys., Inc.*, 558 F.3d 1368, 1377 (Fed. Cir. 2009). A written description that discloses only a certain method does not "necessarily support a broad claim as to every possible type of [method], no matter how different in structure or operation from the inventor's [discussion]." *LizardTech, Inc. v. Earth Resource Mapping, Inc.*, 424 F.3d 1336, 1346 (Fed. Cir. 2005). That is, an inventor's description of one type of method does not entitle the inventor to claim "any and all means for achieving that objective." *Id.*; see also *ICU Medical*, 558 F.3d at 1377–79. Given the disclosure in the specification, and as Google

understands VirtaMove's claim interpretations, the following claim limitations lack adequate written description and are thus invalid:

- **'814 Patent: "**disparate computing environment" (claim 1), "secure, executable, applications related to a service" (claim 1), "a unique root file system that is different from an operating system's root file system" (claim 1), "an operating system's" (claim 1), "including a kernel a set of associated local system files" (claim 1), "the applications each include an object executable by at least some of the different operating systems" (claim 1), "residing permanently" (claim 1), "remain resident" (claim 1), "compatible with a local kernel" (claim 1), "copies or modified copies of the associated local system files that remain resident on the server" (claim 1), "execution file" (claim 2).
- **'058 Patent:** "critical system elements" (claim 1), "OS critical system elements" (claim 1), "shared library critical system elements" (claim 1), "functional replicas of OSCSEs" (claim 1), "some of the SLCSEs stored in the shared library . . . are accessible to some of the plurality of software applications" (claim 1), "forms a part of the one or more of the plurality of software applications" (claim 1), "an instance of a SLCSE provided to at least a first of the plurality of software applications from the shared library is run in a context of said at least first of the plurality of software applications" (claim 1), "a unique instance of a corresponding critical system element" (claim 1), "a SLCSE related to a predetermined function is provided to the first of the plurality of software applications" (claim 1); "a SLCSE for performing a same function is provided to the second of the plurality of software applications for

running a second instance of the SLCSE simultaneously" (claim 1), "unique access to a unique instance of a CSE" (claim 10); "SLCSEs are not copies of OSCSEs" (claim 18).

VI. IDENTIFICATION OF CLAIMS DIRECTED TO INELIGIBLE SUBJECT MATTER UNDER 35 U.S.C. § 101

In addition to the preceding Invalidity Contentions, Google further contends that the following Asserted Claims are drawn to subject matter that is not patentable under 35 U.S.C. § 101:

- **'814 Patent:** Claims 1-5, 10 and 18
- **'058 Patent:** Claims 1, 2, 4, 6, 9, 10, 13–14

The Asserted Patents, both issued before *Alice*, are directed to abstract ideas concerning the functional replicating or copying of system elements or files for use in running applications. The components of the claimed system are nothing more than functional recitations of conventional technology with no new hardware, software, or methods even suggested. These are the types of patent claims that courts routinely invalidate under § 101.

A. INELIGIBLE SUBJECT MATTER

Whether a patent covers patentable subject matter under § 101 is a "threshold" question of law. *Bilski v. Kappos*, 561 U.S. 593, 602 (2010). Thus, "[s]ubject matter eligibility under § 101 may be determined at the Rule 12(b)(6) stage of a case." *ChargePoint, Inc. v. SemaConnect, Inc.*, 920 F.3d 759, 765 (Fed. Cir. 2019). Claim construction is not a prerequisite to a determination under § 101 at the pleadings stage. *Content Extraction & Transmission LLC v. Wells Fargo Bank, Nat. Ass'n*, 776 F.3d 1343, 1349 (Fed. Cir. 2014). *Alice* sets forth a two-step test for patent eligibility under § 101. 573 U.S. at 217-18. First, the Court must determine whether the claims are directed to one of the three patent ineligible concepts under § 101: laws of nature, natural

phenomena, or abstract ideas. *Id*. at 217. If so, the Court must then "consider the elements of each claim both individually and as an ordered combination to determine whether additional elements transform the nature of the claim into a patent-eligible application," which the Supreme Court has described as a "search for an inventive concept—*i.e.*, an element or combination of elements that is sufficient to ensure that the patent in practice amounts to significantly more than a patent upon the ineligible concept itself." *Id.* at 217-18 (cleaned up).

B. THE '058 PATENT IS INVALID UNDER 35 U.S.C. § 101

1. The '058 Patent Is Directed To An Abstract Idea

The '058 Patent is directed to the abstract idea of functionally replicating critical system elements for use with two or more software applications. Claim 1 recites:

1. A computing system for executing a plurality of software applications comprising:

a) a processor;

b) an operating system having an operating system kernel having OS critical system elements (OSCSEs) for running in kernel mode using said processor; and,

c) a shared library having shared library critical system elements (SLCSEs) stored therein for use by the plurality of software applications in user mode and

i) wherein some of the SLCSEs stored in the shared library are functional replicas of OSCSEs and are accessible to some of the plurality of software applications and when one of the SLCSEs is accessed by one or more of the plurality of software applications it forms a part of the one or more of the plurality of software applications,

ii) wherein an instance of a SLCSE provided to at least a first of the plurality of software applications from the shared library is run in a context of said at least first of the plurality of software applications without being shared with other of the plurality of software applications and where at least a second of the plurality of software applications running under the operating system have use of a unique instance of a corresponding critical system element for performing same function, and

iii) wherein a SLCSE related to a predetermined function is provided to the first of the plurality of software applications for running a first instance of the SLCSE, and wherein a SLCSE for performing a same function is provided to the second of the plurality of software applications for running a second instance of the SLCSE simultaneously.

While lengthy, the claim language is recited in highly generic terms, focusing on "an abstract end-result," not "'a specific means or method' for improving technology." *RecogniCorp,*

LLC v. Nintendo Co., 855 F.3d 1322, 1326 (Fed. Cir. 2017) (citation omitted). Rather, claim 1 recites a computer system with generic computer elements such as a processor, operating systems, critical system elements for running in kernel mode, critical system elements for use in user mode, shared libraries, and applications. Nothing in the Amended Complaint suggests otherwise.

Further, the specification concedes that these various generic components all existed in the art. The specification explicitly states that Figure 1 (left) "shows a **conventional architecture"** with these various elements: applications (blue), application libraries (orange), critical system elements (yellow), operating kernel (pink), and hardware (green), where the critical system elements run in kernel mode and the applications operate in user mode. 6:62-7:22. Figure 2(b) (right) shows a "known system architecture" with these same elements, but where some of the critical system elements run in user mode. 7:54-8:3; *see also* Figure 2(a), 7:23-26 ("FIG. 2 *a* shows a system architecture where critical system elements **27***a*, **27***b* execute in user mode but in a distinct context from applications").

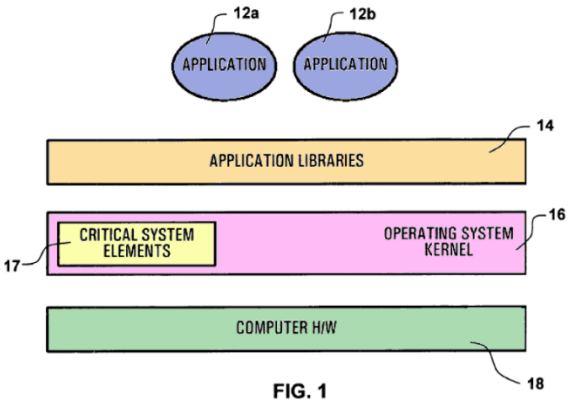


FIG. 1
Prior Art

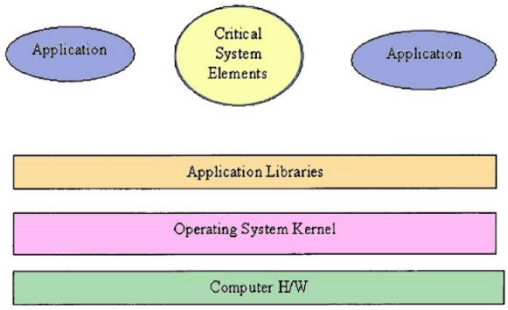


FIG. 2b
Prior Art

Comparing prior art Figures 1 and 2(b) with what the specification indicates is an embodiment of the "invention" in Figure 4 (below) further reveals the abstract nature of Claim 1. Consistent with the abstract idea of functionally replicating critical system elements for use with two or more software applications, Figure 4 consists of the same basic computer elements that the

patent admits were known, e.g., applications, application libraries, critical system elements (including the extended library (8:24-26)), operating system kernel, and hardware.[6] The specification states the purported difference between the arrangement in Figures 1, 2(a), and 2(b) and 4 is that the latter includes "replicate[d]" critical system elements, the "functional replicas" in Claim 1, in the operating system kernel.[7] 7:63-8:3. As the applicant stated in the file history, "the claimed invention utilizes such shared libraries as a way of duplicating critical system elements provided by the operating system." Ex. 1, '058 Prosecution History, Feb. 18, 2009 Amendment at 8. That is the abstract idea to which Claim 1 is directed.

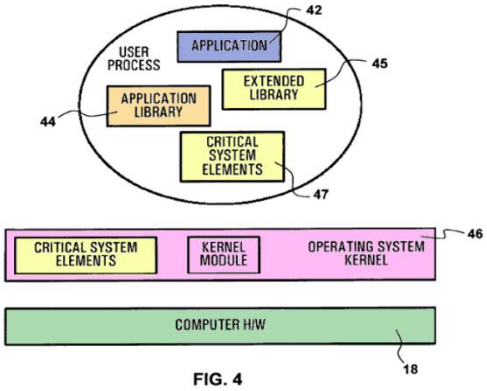


FIG. 4

The claim language recites the various functional elements regarding this abstract process in the claimed "computer system": an operating system that "has" an operating system kernel having OSCSEs, SLCSEs stored in the shared library that "are" functional replicas of OSCSEs and "are" accessible to some of the plurality of software applications, an instance of a SLCSE that "is" run without being shared with other applications, a plurality of software applications running that "have use" of a unique instance of a critical system element, and an SLCSE "provided" to applications for running instances of the SLCSE simultaneously. The claim language, however, does not provide a specific implementation of how these various functions and requirements are

[6] The only specifically recited hardware is "a processor," just a generic part of a computer. *See, e.g., Enco Sys., Inc. v. DaVincia, LLC*, 447 F. Supp. 3d 916, 922-23 (E.D. Mo. 2020), *aff'd Enco Sys., Inc. v. DaVincia, LLC*, 845 F. App'x 953 (Fed. Cir. 2021). And, as noted above, the specification acknowledges there is no difference in the "hardware" of Figure 1 (showing the "prior art") and Figure 4 (showing the "invention").

[7] The critical system elements are in the user process in Figure 4, but again, the specification concedes that was known in the prior art, as shown in Figure 2(b). 7:54-55.

actually accomplished. It just declares they are to be done.

The Federal Circuit has repeatedly held that claims reciting "result-based functional language" for performing computer processes without "sufficiently describ[ing] how to achieve these results" are abstract. *Two-Way Media Ltd. v. Comcast Cable Commc'ns, LLC,* 874 F.3d 1329, 1337 (Fed. Cir. 2017); *see also Interval Licensing*, 896 F.3d at 1345 (claims that "recited only at the broadest, functional level, without explaining how that is accomplished" were abstract); *RecogniCorp, LLC,* 855 F.3d at 1326 (claims that focused on "abstract end-result," not "'specific means or method' for improving technology" were abstract) (citation omitted). Rather, "[t]he purely functional nature of [a] claim confirms that it is directed to an abstract idea, not to a concrete embodiment of that idea." *Affinity Labs of Texas, LLC v. Amazon.com Inc.*, 838 F.3d 1266, 1269 (Fed. Cir. 2016).

For example, in *Eolas Technologies Incorporated v Amazon.com Inc., et. al.*, the Federal Circuit analyzed (under both Step 1 and Step 2) claims which required "an automatically invoked interactive-content application 'configured to operate as part of a distribution application' that 'enable[s] a user' to interact with data objects within a World Wide Web Page." 2024 WL 371959, at *6 (Fed. Cir., Feb. 1, 2024). The Federal Circuit found the claim patent ineligible because "the claims merely describe a desired function or outcome without providing details of the claimed distributed processing." *Id.* The same is true here, where Claim 1 at best recites that certain functionality "is" to be done or "provided," but does not recite the details of how to do so.

Similarly, in affirming the district court's grant of a motion to dismiss, the Federal Circuit in *Aftechmobile Inc. v. Salesforce.com, Inc.* found that, "while the claim recited a computer program to accomplish various functions by running a 'computer program code' stored in a generic computer storage medium and run on a generic computer processor, it nowhere recited *how* the

program code was written or *how* it worked to accomplish those functions." 853 F. App'x 669, 669-70 (Fed. Cir. 2021). And in *Free Stream Media Corp. v. Alphonso Inc*., the claims recited a television, servers, and a mobile device that could "process an embedded object, constrain an executable environment in a security sandbox, and execute a sandboxed application in the executable environment"—but did not sufficiently "identify 'how' [their] functional result is achieved." 996 F.3d 1355, 1359, 1363 (Fed. Cir. 2021) (citation omitted). Thus, despite the patentee's assertion of an "improvement to computer functionality," the claims were directed to "the abstract idea of providing targeted advertising to the mobile device user." *Id.* at 1365; *see also Interval Licensing LLC v. AOL, Inc.,* 896 F.3d 1335, 1339, 1344-45 (Fed. Cir. 2018) (claims abstract where they recited numerous computer steps for coordinating the display of content to a user, including an "attention manager" component for arranging the content, but included no details on "how the attention manager perform[ed] the function[s]" or "*how* to engineer or program the display"; the claims "simply demand[ed] the production of a desired result . . . without any limitation on how to produce that result") (emph. in original). The '058 patent is similarly abstract.

None of the allegations VirtaMove added to the Amended Complaint show otherwise. VirtaMove alleges that the '058 patent is not routine or conventional, citing an April 2009 Gartner "Cool Vendors in Cloud Computing" report from April 2009. Am. Compl., ¶ 25. VirtaMove alleges this visually shows AppZero in its own quadrant in the "Virtualization Landscape," but provides no allegations as to how this April 2009 report, years after the '058 patent's filing, has anything to do with the claimed technology, much less any relevance to the § 101 analysis. And while VirtaMove points to the patent office's findings regarding validity over the prior art (*id.* ¶ 26), the "Supreme Court has unequivocally stated that such inquiries are separate and distinct" from the § 101 analysis. *Miller Mendel, Inc. v. City of Anna, Texas***,** 598 F.Supp.3d 486, 498 (E.D.

Tex. 2022) (citing *Diamond v. Diehr,* 450 U.S. 175, 190 (1981)). Thus, "any argument regarding the alleged novelty or non-obviousness of [a patent] has little, if any, relevance in determining whether [a patent] is directed to patent-ineligible subject matter." *Miller Mendel,* 598 F.Supp.3d at 498. Indeed, "a claim for a *new* abstract idea is still an abstract idea." *Synopsys, Inc. v. Mentor Graphics Corp.*, 839 F.3d 1138, 1151 (Fed. Cir. 2016) (emphasis in original). Also, were it enough that a patent was viewed valid by the PTO over the prior art, a patent could never be held unpatentable under § 101 by a district court.

VirtaMove also argues that the '058 patent is not abstract because it is "inherently rooted in computer technology with no non-technological analog," citing the specification's statement that the technology allows "a CSE to execute in the same context as an application. This then allows, among other things, an ability to deploy multiple instances of a CSE." Am. Compl., ¶ 27 (citing Ex. 3 at 1:46–50). But this is just a desired abstract goal. VirtaMove further alleges "this is despite how 'two software applications that require the same file, yet each requires a different version of the file will conflict,' and despite how 'two applications that require independent access to specific network services will conflict.'" Am. Compl., ¶ 27 (citing Ex. 3 at 1:27–41). It is unclear what VirtaMove is even trying to say here when parsing the specification's description of the prior art, but VirtaMove again provides no connection to the claims. The same is true for VirtaMove's citation to the specification's observation regarding "performance and operational differences" suffered by the prior art. Am. Compl., ¶ 27 (citing Ex. 3 at 1:41–45).

2. <u>The Asserted Patent Claim Adds Nothing Inventive</u>

At *Alice* step two, the Court must determine whether the claims add something significant "apart from" the abstract idea—an inventive concept that "transform[s] the abstract idea . . . into a patent-eligible application." *Chamberlain Grp. v. Techtronic Indus. Co.,* 935 F.3d 1341, 1348–49

(Fed. Cir. 2019). VirtaMove's asserted claim does not. Rather, claim 1 recites only generic computing components, structures, and functions—processor, operating systems, critical system elements running in kernel mode, critical system elements for use in user mode, shared libraries, and applications. Thus, the claimed components and functions are the same "basic functions of a computer" and "purely functional and generic" computer and network components that courts have found merely automate the abstract idea in a "particular technological environment"—which is insufficient to add an inventive concept. *Alice,* 573 U.S. at 225-26; *Free Stream Media,* 996 F.3d at 1366 ("[p]rocessing an 'embedded object' . . . or rendering targeted data 'through a sandboxed application of a mobile device" is non-inventive). "If a claim's only 'inventive concept' is the application of an abstract idea using conventional and well-understood techniques, the claim has not been transformed into a patent-eligible application of an abstract idea." *BSG Tech LLC v. Buyseasons, Inc.,* 899 F.3d 1281, 1290–91 (Fed. Cir. 2018). Even when viewed "as an ordered combination," the claims "ad[d] nothing . . . that is not already present when the steps are considered separately." *Alice,* 573 U.S. at 225 (citation omitted). Indeed, as noted above, the patentee admits that the prior art already established a combination of the known elements, and instead proposes functionally replicating some of those elements. VirtaMove provides no plausible allegations to the contrary in the Amended Complaint.

Further, similar to the claims found unpatentable in *Electric Power Group, LLC v. Alstom SA.,* 830 F.3d 1350, 1355 (Fed. Cir. 2016), Claim 1 "merely call[s] for performance of the claimed . . . functions" without explaining "*how* the desired result is achieved." *Id.; see also TDE Petroleum Data Sols., Inc. v. AKM Enter., Inc.*, 657 F.App'x 991, 993 (Fed. Cir. 2016) (rejecting claims that "recite the ***what*** of the invention, but none of the *how* that is necessary to turn the abstract idea into a patent-eligible application") (italics in original). Thus, this case is unlike those

where a "specific implementation" or "specific improvements" in computer technology are provided that supply an inventive concept. *Bascom Glob. Internet Servs., Inc. v. AT&T Mobility, LLC*, 827 F.3d 1341, 1348-49 (Fed. Cir. 2016). For example, in *Bascom*, the claims were saved at *Alice* step two because the court found the claims recited a "non-generic arrangement" by placing "a filtering tool at a *specific* location" that provided "a technical improvement over prior art ways of filtering such content." *Id.* at 1350. By contrast, here, critical system elements are merely functionally replicated for use with multiple applications.

3. <u>The Dependent Claims Are Similarly Abstract and Not Inventive.</u>

The dependent claims are no less abstract, as they are similarly directed to the abstract idea of using containers with copies of system files for applications so that those applications can be used in different operating environments. The dependent claims also simply recite the use of routine and conventional technology such as shared libraries and system elements (claims 2-5, 10, and 18), operating systems (claim 2), applications (claims 4, 5, and 10), and kernels (claims 3, 4, and 5).

4. <u>VirtaMove Has Not Rebutted Google's Showing</u>

Contrary to VirtaMove's conclusions, Google's proposed abstract idea does not "strip[] away" the character of the claims, nor does Google "ignore[]" the claims or its "wherein requirements." Opp. 14-15. Rather, Google addresses them in detail with supporting case law showing them to be abstract (Dkt. 36, 9-14), which VirtaMove simply ignores. VirtaMove also does not confront the admissions in the specification regarding the existence of the various claim elements in the art. *Id*. It also ignores the admissions in the specification and prosecution history that the claims are directed to "replicating" or "duplicating" critical system elements, which is exactly what Google's abstract idea recites. *Id.*

VirtaMove argues that while "Google points to 'processors' and other specific elements, it ignores the programming claimed for that hardware…." Opp. 15. But VirtaMove does not dispute that the hardware claimed is generic and known to the art. Dkt. 36, 9-11. Nor does VirtaMove point to any *claimed* "programming"; Claim 1 is again recited in a purely functional manner as Google demonstrated, and VirtaMove also fails to rebut. Dkt. 36, 11-13 (collecting cases). VirtaMove also makes its own proposal as to what the claims are "directed to":

> A computing system wherein some shared library critical system elements (SLCSEs) are functional replicas of operating system critical system elements (OSCSEs) and where at least one SLCSE related to a predetermined function is provided to a first application for a first instance of the SLCSE and a SCLCSE for performing the same function is provided to a second application for running a second instance of the SLCSE simultaneously.

Opp. 14. Although this has more words than Google's abstract idea, VirtaMove's recasting is no less abstract. Indeed, it concedes that the Claim 1 is directed to making functional replicas of critical system elements for use with two or more applications, just like Google's proposed abstract idea. All VirtaMove seems to add is the abstract notion that a critical system component relates to a predetermined function, at best, just making Google's point as to Step 1.

Further, VirtaMove again points to some unclaimed, unidentified "programming" that is supposedly inventive. Opp. 15. VirtaMove also contends Google does not focus on the "ordered combination" of elements, but VirtaMove says nothing as to just what ordering or combination it is referring to, much less how it is supposedly "inventive" to pass muster on Step 2. *Id.* VirtaMove also oddly discusses two ***Google*** patents. *Id*. VirtaMove's argument appears to be that because Google has two patents that make mention of "containers," without regard for whatever these patents actually claim, this shows that the '058 patent, which does not even claim the use of "containers," somehow passes muster on Step 2. This shows nothing.

C. THE '814 PATENT IS INVALID UNDER 35 U.S.C. § 101

Here too, the Amended Complaint only specifically asserts infringement of Claim 1 of the '814 Patent (Compl. Ex. 2), and Google, therefore, treats Claim 1 as representative.

1. The '814 Patent Is Directed To An Abstract Idea

The '814 Patent is directed to the abstract idea of using containers with copies of system files for applications, so that those applications can be used in different operating environments. Claim 1 recites:

> 1. In a system having a plurality of servers with operating systems that differ, operating in disparate computing environments, wherein each server includes a processor and an operating system including a kernel a set of associated local system files compatible with the processor, a method of providing at least some of the servers in the system with secure, executable, applications related to a service, wherein the applications are executed in a secure environment, wherein the applications each include an object executable by at least some of the different operating systems for performing a task related to the service, the method comprising:
>
> storing in memory accessible to at least some of the servers a plurality of secure containers of application software, each container comprising one or more of the executable applications and a set of associated system files required to execute the one or more applications, for use with a local kernel residing permanently on one of the servers;
>
> wherein the set of associated system files are compatible with a local kernel of at least some of the plurality of different operating systems, the containers of application software excluding a kernel,
>
> wherein some or all of the associated system files within a container stored in memory are utilized in place of the associated local system files that remain resident on the server,
>
> wherein said associated system files utilized in place of the associated local system files are copies or modified copies of the associated local system files that remain resident on the server,
>
> and wherein the application software cannot be shared between the plurality of secure containers of application software,
>
> and wherein each of the containers has a unique root file system that is different from an operating system's root file system.

As with '058 claim 1, while lengthy, claim 1 of the '814 Patent is recited in highly generic terms, focusing on "an abstract end-result," not "'a specific means or method' for improving technology."

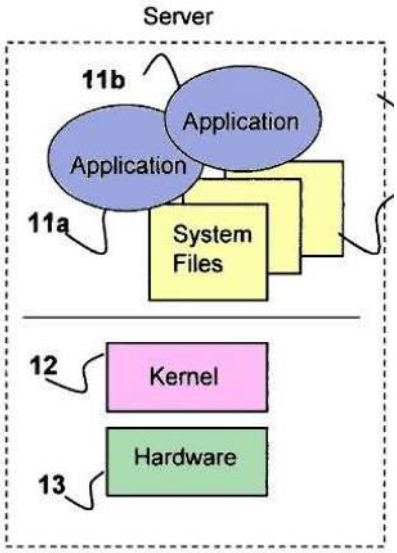
RecogniCorp, LLC, 855 F.3d at 1326. Indeed, claim 1 similarly recites a computer system with generic computer elements such as containers,[8] servers, operating systems, computing environments, a processor, a kernel, local system files, applications, objects, and a root file system. These are many of the same and/or similar elements of the '058 Patent, which actually incorporates the provisional application of the '814 Patent by reference. Ex. 3 at 8:7-10. Both patents stem from provisional applications that were filed in September 2003, and both patents were filed in September 2004.

As with the '058 Patent, the '814 specification acknowledges the existence of these various conventional elements in the "Background of Invention" section. 1:20-3:19. Indeed, the specification lists examples of existing system files and libraries that are to be copied for the claimed invention. 2:53-3:19. Figure 1 (below) illustrates "a containerized system in accordance with an embodiment of this invention." 6:10-11. Like Figure 4 of the '058 Patent, the components shown in this figure are purely conventional; applications, system files, kernels, and hardware were all known in the art. The specification indicates that the system files used in the container (shown above the horizontal line) are used "in place of system files such as library and configuration files present [sic] typically used in conjunction with the execution of application." 6:59-7:3. Consistent with the abstract idea of using containers with copies of operating system files for applications so applications can be used in different operating environments, in the '814 Patent, these known basic computer elements have simply been copied into a container so they can run with more than one application. 10:18-11:7; Figure 3. As the applicant similarly noted in a September 3, 2008 response to an Office Action, "Applicants' claimed invention is defined as having 'secure' containers of

[8] VirtaMove alleges it is an "an innovator and pioneer in containerization" (Am. Compl., ¶ 3), but does not, and cannot, contend it (or the '814 patent) invented "containers."

application software which include OS files that are copies or modified copies of OS files while preserving the original OS files by 'not' overwriting them." Ex. 2, '814 Patent, Sept. 3, 2008 Amendment at 15-16.

Also similar to the '058 Patent, Claim 1 of the '814 Patent recites various functional elements of this process in the claimed computer system: "storing" containers of application software with one or more of the executable applications and system files, the set of associated system files "are compatible" with a local kernel of at least some of the plurality of different operating systems, the containers of application software exclude a kernel, some or all of the copied associated system files "are utilized in place of" system files resident on the server, application software "cannot be shared" between the plurality of secure containers of application software, and containers have a unique root file system. But as with the '058 Patent, Claim 1 of the '814 Patent does not provide the actual implementation details of these functions and requirements. It simply declares they are to be done. And here too, the claims do not require any particular hardware. The only recited hardware is generic "servers," "memory," and a "processor." VirtaMove does not even try to allege otherwise in its Amended Complaint. As detailed in Section III(A), *supra* (collecting cases), the Federal Circuit has consistently found such claims abstract (and ineligible) where "[t]he purely functional nature of [a] claim confirms that it is directed to an abstract idea, not to a concrete embodiment of that idea." *Affinity Labs II*, 838 F.3d at 1269.



Only further showing the abstract nature of both patents, VirtaMove adds largely the same allegations to the Amended Complaint for the '814 as it did for the '058, claiming it too is directed

to the "problem of application portability across operating systems." This time, VirtaMove alleges the '814 is not abstract because it is directed to the "problem of application portability across operating systems [that] is inherently rooted in computer technology with no non-technological analog," citing the specification's statement that the "invention provides a solution whereby a plurality of services can conveniently be installed on one or more servers in a cost effective and secure manner." Am. Compl., ¶ 15 (citing Ex. 1 at 2:13–15). But, here too, this is just a desired abstract goal. VirtaMove also states "This is despite how 'a collection of applications . . . must be separated with each application installed on an individual computer system,' and despite how 'certain applications require a specific version of operating system facilities and as such will not co-exist with applications that require another version.'" *Id.* (citing Ex. 1 at 1:27–41). It is unclear what this statement even means, much less what it has to do with the patent claims. VirtaMove's further argument that the claimed technology only requires "one operating system," only further underscores its abstract, result-based nature. *Id.*

2. <u>The Asserted Patent Claim Adds Nothing Inventive</u>

Here too, claim 1 of the '814 Patent recites generic computing components, structures and functions—such as servers, operating systems, computing environments, a processor, a kernel, local system files, applications, objects, containers, and a root file system. It lacks any technical specificity and provides no mechanism for how to perform the claimed functional steps. Rather, as discussed above, the specification only discloses technologies that it admits were well-understood, routine, and conventional. Again, the claimed components and functions are the same sort of "basic functions of a computer" and "purely functional and generic" computer and network components that courts have found merely automate the abstract idea in a "particular technological environment"—which is insufficient to add an inventive concept as a matter of law. *See* Section

III(B), *supra.* VirtaMove again provides no plausible allegations to the contrary in the Amended Complaint. The ’814 Patent is patent ineligible and should be dismissed.

3. <u>The Dependent Claims Are Similarly Abstract and Not Inventive.</u>

The dependent claims are no less abstract, as they are similarly directed to the same abstract idea of functionally replicating critical system elements for use with two or more software applications. The dependent claims also simply recite the use of routine and conventional technology such as execution files (claim 2), applications and system files (claims 4, 10, 13, and 14), IP address, host name, and MAC address (claim 6), CPU memory, network bandwidth, and disk allocation (claim 9), memory (claim 14), and servers (claim 14).

4. <u>VirtaMove Has Not Rebutted Google’s Showing</u>

Here too, Google does not “strip[] away” the character of Claim 1 or ignore the claims in formulating its proposed abstract idea. Opp. 9. Rather, Google addresses the claim language and the intrinsic evidence in detail in its Motion. Dkt. 36, 16-19. Yet, again, VirtaMove largely ignores Google’s arguments regarding the claims, admissions in the intrinsic evidence, and Google’s cited case law.

Rather, VirtaMove argues that “using containers with copies of system files for applications, so that those applications can be used in different operating environments” is somehow not abstract because it relates to what VirtaMove concludes to be “computer-specific” problems or solutions. Opp., 8. But “applications [being] used in different operating environments” is simply an abstract goal of executing the functional, abstract idea of “using containers with copies of system files for applications,” and does not demonstrate any specific and non-abstract step(s) recited within the claims *Elec. Power Grp. LLC v. Alstom S.A*, 830 F.3d 1350, 1356 (Fed. Cir. 2016) (“[T]he essentially result-focused, functional character of claim language

has been a frequent feature of claims held ineligible under § 101."). VirtaMove does cite cases that it indicates have "analogous" claims, but here too, VirtaMove does not even try to analogize the claims in these cases to Claim 1. Opp. 8-9. It just repeats that the courts in those cases found the claims in ***those*** cases were directed to non-abstract computer improvements. *Id.* But that does nothing to show that ***Claim 1*** is not abstract. It is. VirtaMove cites no case, as it cannot, that the '814 Patent is not abstract simply because it relates to computers. Opp. 8.

VirtaMove also points to the PTO's findings regarding validity over the prior art. Opp. 8. But as Google has already shown, such inquiries under §§ 102 and 103 are "separate and distinct" from the § 101 analysis. Dkt. 36, 13 (collecting cases). VirtaMove ignores this too. VirtaMove again makes its own proposal as to what the claims are "directed to":

> storing a plurality of secure containers of application software; wherein associated system files utilized in place of the associated local system files are copies or modified copies of the associated local system files resident on the server, and each of the containers has a unique root file system.

Opp. 6-7, *see also* 9-10. Yet, here too, while longer than Google's abstract idea, VirtaMove's recasting is no less abstract. Indeed, it concedes that Claim 1 is directed to using containers with copies of system files for applications. Beyond that, all it seems to add is the abstract notion of "uniqueness" to the element of a root file system, which was undisputedly known in the prior art. Here too, VirtaMove's characterization simply makes Google's point.

VirtaMove further again points to some unclaimed, unidentified "programming" that is supposedly inventive. Opp. 10. VirtaMove also says Google does not focus on the "ordered combination" of elements, but VirtaMove says nothing as to just what ordering or combination it is referring to, much less how it is supposedly "inventive" to pass muster on Step 2. *Id.* Nor does VirtaMove dispute that the claimed hardware elements are conventional. Dkt. 36, 14. VirtaMove's argument that an invention's "compatibility with conventional systems *does not* render it

ineligible" (Opp. 11) is a red herring. That is not Google's argument.

VirtaMove cites *BASCOM* and *DDR Holdings* as supposedly supporting that Claim 1 has an inventive step (Opp. 10-11), but as with the cases it discusses for Step 1, VirtaMove makes no attempt to show the claims at issue in those cases are at all analogous to Claim 1 here. VirtaMove also contends that Google "never deals with the facts, evidence, and specific allegations in the intrinsic record and the FAC." Opp. 11. This is untrue. Google addresses this in detail in its Motion. Dkt. 36, 9-11, 13-15. While VirtaMove repeats what it says regarding the April 2009 Gartner Report in its Amended Complaint (Opp. 12), VirtaMove ignores what Google states in rebuttal. Dkt. 36, 13. Nor does the reference to "containers," as relating to cloud services, on a Google website somehow show the '814 Patent passes muster under Step 2 as VirtaMove appears to suggest. Opp. 12.

Google further incorporates by reference its briefing on in its motions to dismiss. Dkt. 21 at 9-19, Dkt. 36 at 8-20; Dkt. 47 at 5-9.

Google reserves the right to identify additional grounds of invalidity under Section 101 in response to Plaintiff's arguments and positions, any amendments to Plaintiff's Infringement Contentions, the Court's claim construction, or information uncovered during further discovery.

VII. DOCUMENT PRODUCTION ACCOMPANYING PRELIMINARY INVALIDITY CONTENTIONS

Google is producing and/or making available for inspection documents required under the parties' Scheduling Order (Dkt. 34), including all prior art referenced above and technical documents sufficient to show the operation of the accused products at the following Bates numbers: GOOG-VIRTA-00000818-25274 and GOOG-VIRTA-PA-00000001-2848. Further, Google has investigated and identified relevant source code sufficient to show the operation of the accused products. Google will make this source code available for inspection once the parties

submit and the Court enters a Protective Order that includes protections for Google's Confidential Source Code Information.

Google reserves the right to identify and produce additional documents pursuant to OGP § I and the orders of the Court.

DATED: October 17, 2024

/s/ David A. Perlson

David A. Perlson (admitted pro hac vice)
davidperlson@quinnemanuel.com
QUINN EMANUEL URQUHART & SULLIVAN, LLP
50 California Street, 22nd Floor
San Francisco, California 94111-4788
Telephone: (415) 875-6600
Fax: (415) 875-6700

Deepa Acharya
deepaacharya@quinnemanuel.com
QUINN EMANUEL URQUHART & SULLIVAN, LLP
1300 I Street NW, Suite 900
Washington, D.C. 20005
Telephone: (202) 538-8000
Fax: (202) 538-8100

Katharine Lee Carmona (TX SBN 00787399)
kcarmona@jw.com
JACKSON WALKER L.L.P.
100 Congress Avenue, Suite 1100
Austin, Texas 78701
(512) 236-2000
(512) 236-2002 (facsimile)

Erica Benites Giese (TX SBN 24036212)
egiese@jw.com
JACKSON WALKER L.L.P.
1900 Broadway, Suite 1200
San Antonio, Texas 78215
(210) 978-7700
(210) 978-7790 (facsimile)

Nathaniel St. Clair, II (TX SBN 24071564)
nstclair@jw.com
JACKSON WALKER L.L.P.

2323 Ross Avenue, Suite 600
Dallas, Texas 75201
(214) 953-6000
(214) 953-5822 (facsimile)

Attorneys for Defendant Google LLC

CERTIFICATE OF SERVICE

I hereby certify that on October 17, 2024, the foregoing was served via email on all counsel of record.

/s/ Nicola R. Felice