

Petition for *Inter Partes* Review of U.S. Patent No. 7,519,814
Case IPR2025-00599

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

INTERNATIONAL BUSINESS MACHINES CORP.,

Petitioner

v.

VIRTAMOVE, CORP.,

Patent Owner

Case IPR2025-00599

U.S. PATENT NO. 7,519,814

**PETITION FOR *INTER PARTES* REVIEW
UNDER 35 U.S.C. § 312 AND 37 C.F.R. § 42.104**

TABLE OF CONTENTS

I.	INTRODUCTION	1
II.	BACKGROUND OF THE '814 PATENT	2
A.	Claims.....	3
B.	Overview of the '814 Patent.....	3
C.	Prosecution History of the '814 Patent	4
III.	IDENTIFICATION AND BASIS OF CHALLENGE	4
IV.	OVERVIEW OF THE ASSERTED PRIOR ART	9
A.	McMillan	9
B.	Schaefer	12
C.	Gaines	14
D.	Tormasov	14
V.	LEVEL OF ORDINARY SKILL IN THE ART	15
VI.	CLAIM CONSTRUCTION	16
VII.	SPECIFIC GROUNDS FOR PETITION	18
A.	Ground I: McMillan Alone or Alternatively In View of Gaines and/or Tormasov.....	18
1.	Claim 1	18
2.	Claim 2	34
3.	Claim 6	35
4.	Claim 9	40
5.	Claim 10	41

Petition for *Inter Partes* Review of U.S. Patent No. 7,519,814
Case IPR2025-00599

B.	Ground II: Schaefer Alone or in View of Gaines and/or Tormasov	42
1.	Claim 1	42
2.	Claim 2	58
3.	Claim 6	58
4.	Claim 9	61
5.	Claim 10	62
VIII.	SECONDARY CONSIDERATIONS	63
IX.	PTAB DISCRETION SHOULD NOT PRECLUDE INSTITUTION	64
A.	Discretionary Denial Under 35 U.S.C. § 325(d) is Not Warranted	64
B.	Discretionary Denial Under 35 U.S.C. § 314(a) is Not Warranted	65
X.	MANDATORY NOTICES	66
A.	Real Party-In-Interest	66
B.	Related Matters	66
C.	Counsel Information	67
D.	Service Information	68
XI.	PAYMENT OF FEES	68
XII.	CERTIFICATION	68
XIII.	CONCLUSION	69

TABLE OF AUTHORITIES

Cases	Page(s)
<i>10X Genomics, Inc. v. Bio-Rad Laboratories, Inc.</i> , IPR2020-00089, Paper 8 (PTAB Apr. 27, 2020)	18
<i>Advanced Bionics, LLC v. Med-El Elektromedizinische Geräte GmbH</i> , IPR2019-01469, Paper 6 (PTAB Feb. 13, 2020).....	64
<i>Amazon.com, Inc. v. VirtaMove, Corp.</i> , IPR2025-00566, IPR2025-00563	67
<i>Apple Inc. v. Fintiv, Inc.</i> , IPR2020-00019, Paper 11 (P.T.A.B. Mar. 20, 2020).....	64, 66
<i>Becton, Dickinson, & Co. v. B. Braun Melsungen AG</i> , IPR2017-01586, Paper 8 (PTAB Dec. 15, 2017)	64
<i>Bowtech Inc. v. MCP IP, LLC</i> , IPR2019-00383, Paper 14 (PTAB Aug. 6, 2019).....	64
<i>CommScope Techs. LLC v. Dali Wireless, Inc.</i> , IPR2022-01242, Paper 23 (Feb. 27, 2023)	66
<i>Fasteners for Retail, Inc. v. RTC Indus., Inc.</i> , IPR2019-00994, Paper 9 (PTAB Nov. 5, 2019).....	64
<i>Google, LLC v. VirtaMove, Corp.</i> , IPR2025-00488, IPR2025-00487	67
<i>Phillips v. AWH Corp.</i> , 415 F.3d 1303 (Fed. Cir. 2005) (en banc)	16
<i>Red Hat, Inc. v. VirtaMove, Corp.</i> , No. 3:24-cv-04740 (N.D. Cal.).....	67
<i>Sotera Wireless, Inc. v. Masimo Corp.</i> , IPR2020-01019, Paper 12 (PTAB Dec. 1, 2020)	65
<i>Thryv, Inc v. Click-To-Call Techs., LP</i> , 140 S. Ct. 1367 (2020).....	66

Petition for *Inter Partes* Review of U.S. Patent No. 7,519,814

Case IPR2025-00599

VirtaMove, Corp. v. Amazon.com, Inc.,

No. 7:24-cv-00030 (W.D. Tex.) 66, 67

VirtaMove, Corp. v. Google LLC,

No. 7:24-cv-00033 (W.D. Tex.) 66, 67

VirtaMove, Corp. v. Hewlett Packard Enterprise Co.,

No. 2:24-cv-00093 16, 66

VirtaMove, Corp. v. International Business Machines Corp.,

No. 2:24-cv-00064-JRG (E.D. Tex.) 16, 63, 65

VirtaMove, Corp. v. Microsoft Corp.,

No. 7-24-cv-00338 (W.D. Tex.) 66

VirtaMove, Corp. v. Oracle Corp.,

7-24-cv-00339 (W.D. Tex.) 67

Western Digital Corp. v. SPEX Techs., Inc.,

IPR2018-00084, Paper 14 (PTAB Apr. 25, 2018) 17

Wyers v. Master Lock Co.,

616 F.3d 1231 (Fed. Cir. 2010) 63

Statutes

35 U.S.C. § 102 4, 5

35 U.S.C. § 103 5

35 U.S.C. § 119 5

35 U.S.C. § 282(b) 16

35 U.S.C. § 301(a)(2) 17

35 U.S.C. § 314(a) 64, 65

35 U.S.C. § 315(e)(1) 68

35 U.S.C. § 325(d) 64

Rules

Rule 42.104(a).....	68
---------------------	----

Other Authorities

37 C.F.R. § 42.10(b)	68
37 C.F.R. § 42.15(a)(1).....	68
37 C.F.R. § 42.100(b) (2018).....	16

LIST OF EXHIBITS

Exhibit No.	Description
1001	U.S. Patent No. 7,519,814 (the “814 Patent”)
1002	File History of U.S. Pat. No. 7,519,814
1003	Declaration of Todd C. Mowry Ph.D. in Support of Petition for <i>Inter Partes</i> Review of U.S. Patent No. 7,519,814
1004	<i>Curriculum Vitae</i> of Todd C. Mowry Ph.D.
1005	U.S. Patent No. 7,673,308 (“McMillan”)
1006	U.S. Provisional Application No. 60/427,339 (the “’339 application”)
1007	U.S. Patent No. 5,961,582 (“Gaines”)
1008	U.S. Patent Pub. No. 2002/0174215 A1 (“Schaefer”)
1009	ANDREW S. TANENBAUM, MODERN OPERATING SYSTEMS (2d ed. 2001)
1010	D. A. Solomon, “The Windows NT kernel architecture,” in <i>Computer</i> , vol. 31, no. 10, pp. 40-47, Oct. 1998
1011	U.S. Patent Pub. No. 2002/0124072 A1 (“Tormasov”)
1012	U.S. Patent Pub. No. 2003/0014466 A1 (“Berger”)
1013	File History of European Patent Application No. 04021916.4 (“EP FH”)
1014	Redline between the ’339 application and McMillan
1015	Dkt. 124 Joint Claim Construction and Prehearing Statement
1016	Plaintiff’s Third Supplemental Objections and Responses to Defendants’ First Set of Common Interrogatories (Nos. 1-9). <i>VirtaMove, Corp. v. Hewlett Packard Enterprise Co.</i> , 2:24-cv-00093 (Nov. 18, 2024 E.D. Tex.)
1017	Declaration of Leslie Hesdorfer
1018	IEEE Declaration of Gordon MacPherson
1019	U.S. Patent No. 5,724,510 (“Arndt”)
1020	U.S. Patent No. 7,028,305

LIST OF CHALLENGED CLAIMS

Reference Number	Claim Limitation
Claim 1	
1 [Pre.a]	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 [and] a set of associated local system files compatible with the processor,
1 [Pre.b]	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,
1 [Pre.c]	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:
1[a]	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;
1[b]	wherein the set of associated system files are compatible with a local kernel of at least some of the plurality of different operating systems,
1[c]	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,
1[d]	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,

1[e]	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.
Claim 2	
2	A method as defined in claim 1, wherein each container has an execution file associated therewith for starting the one or more applications.
Claim 6	
6	A method as defined in claim 2, comprising the step of assigning a unique associated identity to each of a plurality of the containers, wherein the identity includes at least one of IP address, host name, and MAC address.
Claim 9	
9	A method as defined in claim 2, wherein server information related to hardware resource usage including at least one of CPU memory, network bandwidth, and disk allocation is associated with at least some of the containers prior to the applications within the containers being executed.
Claim 10	
10	A method as defined in claim 2, wherein in operation when an application residing within a container is executed, said application has no access to system files or applications in other containers or to system files within the operating system during execution thereof.

International Business Machines Corp. (“IBM” or “Petitioner”) requests *inter partes* review of Claims 1, 2, 6, 9, and 10 (“Challenged Claims”) of U.S. Patent No. 7,519,814 (the “’814 Patent”) (Ex-1001).¹

I. INTRODUCTION

The ’814 Patent concerns a computer system architecture utilizing software containers accessible to multiple standalone servers, where at least two of the servers have different operating systems, such as different instances of the same operating system version (*e.g.*, two instances of Windows 2000); different versions of the same operating system (*e.g.*, Windows 2000 and Windows NT), or completely distinct operating systems (*e.g.*, Windows and Unix). These containers include application(s) and some system files that allow them to run independently of other containers while using the kernel of the underlying operating system (OS) for access to hardware and other system resources. In this way, different applications can purportedly run independently at the same time without competing for the same system resources.

But this use of containers was already well-known in the art before the alleged invention of the ’814 Patent. For example, McMillan, which is relied upon in this Petition and was not considered during original prosecution of the ’814 Patent,

¹ All emphasis in quotations is added unless otherwise noted. All citations herein are exemplary and not exhaustive.

discloses and renders obvious the claimed containerization technology. Other invalidating references are also presented below, including the Schaefer reference that formed the basis of an insurmountable rejection to the '814 Patent's counterpart in the European Patent Office, but was not substantively discussed or relied upon during original prosecution in the USPTO. Accordingly, Petitioner respectfully requests that Challenged Claims be cancelled as unpatentable for at least the reasons detailed below.

II. BACKGROUND OF THE '814 PATENT

The '814 Patent issued on April 14, 2009, from U.S. Application No. 10/939,903 (Ex-1002, "the '903 Application") filed on September 13, 2004. The '903 Application claims priority to provisional application No. 60/512,103 filed October 20, 2003, and provisional application No. 60/502,619 filed September 15, 2003. Without admitting that the claims of the '814 Patent are entitled to a priority date of September 15, 2003, Petitioner assumes a September 15, 2003 priority date for purposes of this Petition.²

² While Patent Owner ("PO") has alleged an earliest priority date of September 18, 2002 in pending district court litigation (*see* Section X.B. below) based on alleged prior invention, PO has not sufficiently explained or corroborated the factual basis for such the alleged priority date. Petitioner reserves the right to challenge this or any other alleged priority date that PO may raise in response to this Petition.

A. Claims

The '814 Patent has 34 claims. Claims 1, 2, 6, 9, and 10 are challenged herein, of which claim 1 is independent.

B. Overview of the '814 Patent

The '814 Patent “relates to computer software” and “[i]n particular. . . relates to management and deployment of server applications.” Ex-1001, 1:14-16. The '814 Patent recognizes that “[i]t is traditionally the task of an operating system to provide a mechanism to safely and effectively control access to shared resources required by application programs. This is the foundation of multi-tasking systems that allow multiple disparate applications to co-exist on a single computer system.” Ex-1001, 1:20-26. The '814 Patent notes, however, that “[t]he current state of the art creates an environment where a collection of applications, each designed for a distinct function, must be separated with each application installed on an individual computer system.” *Id.*, 1:27-30. The '814 Patent explains that this separation requirement is necessitated “by conflicts over shared resources, such as network port numbers that would otherwise occur,” “the requirement to securely separate data such as files contained on disk-based storage and/or applications between disparate users,” or “the reality that certain applications require a specific version of operating system facilities and as such will not co-exist with applications that require another version.” *Id.* at 1:31-40.

The '814 Patent purports to solve this problem of application separation through the use of separate software “containers,” where a container is a group of files that includes at least one application and system files needed for the container to interact with the underlying OS kernel. As discussed below, however, this containerization solution was already well known in the art and taught by references such as McMillan and Schaefer.

C. Prosecution History of the '814 Patent

The application that issued as the '814 Patent was originally filed with 34 claims, two of which were independent. Ex-1002 (FH), 34-39. During prosecution, the application was rejected under 35 U.S.C. §102(b) in view of Forbes. Ex-1002 (FH), 237-274. The applicant traversed the rejection by amending the claims. *See* Ex-1002 (FH), 287-288. The Examiner subsequently issued a Notice of Allowance. Ex-1002 (FH), 339-347. As discussed below, all features allegedly missing from the prior art during original prosecution are taught by the prior art discussed below.

III. IDENTIFICATION AND BASIS OF CHALLENGE

Petitioner's challenge is based on the following prior art:

- **McMillan (Ex-1005)** – U.S. Patent No. 7,673,308, filed November 18, 2003 and issued March 2, 2010, which claims priority to U.S. Provisional Application No.

60/427,339 (“’339 application”) filed on November 18, 2002. McMillan is prior art at least under 35 U.S.C. § 102(e).³

- **Schaefer (Ex-1008)** – U.S. Patent Pub. No. 2002/0174215 A1 (“Schaefer”), filed on May 16, 2001 and published on November 21, 2002, which is prior art under at least under 35 U.S.C. §§ 102(a) and (e).
- **Gaines (Ex-1007)** – U.S. Patent No. 5,961,582 (“Gaines”), filed on October 25, 1994 and issued on October 5, 1999, which is prior art under at least 35 U.S.C. §102 (b).
- **Tormasov (Ex-1011)** – U.S. Patent Pub. No. 2002/0124072 A1 (“Tormasov”), filed on July 30, 2001 and published on September 5, 2002, which is prior art at least under 35 U.S.C. §§ 102(a), (b) and (e).

As described in the below chart and in the accompanying Declaration of Dr. Mowry, the ’339 application’s specification is nearly identical to McMillan’s specification as issued, McMillan’s claims are supported by the ’339 application, and thus McMillan is prior art to the ’814 Patent. Ex-1003, ¶23. McMillan was filed within one year of the ’339 application’s filing, names at least one inventor in common, and specifically references the ’339 application. Ex-1005, Cover; Ex-

³ Cites to 35 U.S.C. §§ 102, 103, and 119 are to the pre-AIA versions applicable here.

1006, 11. A redline between McMillan and the '339 application is provided as Ex-1014:

McMillan Claim 1 Element	Exemplary McMillan Provisional Disclosure (Ex-1006)	Corresponding McMillan Patent Disclosure
A computing machine having a computing architecture, comprising:	Ex-1006 at 1, Figs. 1-4 (“This invention is directed to the creation of multiple semi-independent virtual OS environments within a single operating system (OS)”).	Ex-1005, at Figs. 1-4, 1:12-14, 2:10-12, 2:55-57.
a base operating system (OS) installed to the computing machine, the base OS having a base OS file system and a base OS registry;	Ex-1006 at 1, Figs. 1-2 (“This invention is directed to the creation of multiple semi-independent virtual OS environments within a single operating system (OS) The file system and registry information for each environment is independent of the base OS and other environments. . . . In addition, all of the applications run on a single OS desktop”) <i>See also</i> at 2-3.	Ex-1005, at Figs. 1-2, 1:12-14, 2:10-19, 2:55-64, 3:1-21, 3:57-59.
at least one virtual OS environment within the base OS, the virtual OS environment having a virtual OS file system and a virtual OS registry which are independent of the base OS file system and the base OS registry, wherein a dynamic link library (DLL) is injected into an application running under the virtual OS	Ex-1006 at 1, Figs. 1-2 (“This invention is directed to the creation of multiple semi-independent virtual OS environments within a single operating system (OS). . . . In this way each virtual OS environment appears to be an independent OS for the applications running within it. The file system and registry information for each environment is independent of the base OS and other environments. . . . ”), 2 (“To create the virtual OS environments , the OS APIs that access the file system and registry directly and indirectly must be changed to redirect these accesses into the virtual OS environments file system and registry .”).	Ex-1005, at Figs. 1-2, 1:12-14, 2:10-19, 2:55-64, 3:1-21, 3:49-4:32, 4:57-65.

environment, the DLL being programmed to	<i>This is done by injecting a DLL into every application that is executed.”); see also id. at 3-4.</i>	
determine that the application should be run under the virtual OS environment instead of the base OS;	Ex-1006 at 3, Fig. 2 (“As mentioned earlier, not all applications are run from virtual OS environments and different applications may be run from different virtual OS environments. <i>When the injected DLL is first loaded it must determine which of the virtual OS environments (if any) to redirect to. . . . The base OS contains a list of directories where applications should be redirected to specific virtual OS environments.”); see also id. at 4, 6.</i>	Ex-1005, at Fig. 2, 3:17-21, 3:49-4:32, 4:57-65.
call a function of the injected DLL rather than calling an API of the base OS;	Ex-1006 at 2, Figs. 3-7 (“This is done by injecting a DLL into every application that is executed. . . . When the application (or one of the applications DLLs) attempts to call an OS API (directly or indirectly) that will access the file system or registry, <i>the injected DLL’s function is called instead.”); see also id. at 3-7.</i>	Ex-1005, at Figs. 3-7, 3:17-37, 3:49-5:8, 5:20-37.
modify at least one parameter from a calling function of the application to direct the at least one parameter to a location of the virtual OS environment;	Ex-1006 at 2, Figs. 3-7 (“The injected DLL’s function then examines <i>the parameters that are passed to the API and modifies them</i> to direct them into the virtual OS environment location instead.”); see also id. at 3-7.	Ex-1005, at Figs. 3-7, 3:34-39, 3:49-4:56, 4:66-5:8, 5:20-37.
after modifying the at least one parameter, call the API of the base OS with the at least one modified parameter;	Ex-1006 at 2, Figs. 3-7 (“The injected DLL’s function then examines the parameters that are passed to the API and modifies them <i>The injected DLL’s function then calls the real OS API with the modified parameters to perform the required function.”); see also id. at 3-7.</i>	Ex-1005, at Fig. 3-7, 3:34-4:56, 4:66-5:8, 5:20-37.

receive information from the API of the base OS;	Ex-1006 at 2-3, Figs. 3-7 (“ <i>Lastly, the injected DLL’s function returns to the calling function, returning any information from the OS API function.</i> ”); This returned information is also modified to convert the file system or registry location information back from the virtual OS environment location.”); <i>see also id.</i> at 3-7.	Ex-1005, at Figs. 3-7, 3:34-4:56, 4:66-5:8, 5:20-37.
modify the information from the API of the base OS to convert the information back from a virtual OS environment location;	Ex-1006 at 2-3, Figs. 3-7 (“Lastly, the injected DLL’s function returns to the calling function, returning any information from the OS API function. <i>This returned information is also modified to convert the file system or registry location information back from the virtual OS environment location.</i> ”); <i>see also id.</i> at 3-7.	Ex-1005, at Figs. 3-7, 3:34-4:56, 4:66-5:8, 5:20-37.
return the information to the calling function of the application.	Ex-1006 at 2-3, Figs. 3-7 (“ <i>Lastly, the injected DLL’s function returns to the calling function, returning any information from the OS API function.</i> ”); This returned information is also modified to convert the file system or registry location information back from the virtual OS environment location. By using this method, the injected DLL functions can execute code both before and after the OS API function.”); <i>see also id.</i> at 3-7.	Ex-1005, at Figs. 3-7, 3:34-4:56, 4:66-5:8, 5:20-37.

Petitioner requests cancellation of the Challenged Claims on the grounds below:

Ground	Challenged Claims	Proposed Statutory Rejection
I	Claims 1, 2, 6, 9 and 10	Obvious under §103 in view of McMillan, or alternatively in view of McMillan in combination with Gaines and/or Tormasov.

II	Claims 1, 2, 6, 9 and 10	Obvious under §103 in view of Schaefer, or alternatively in view of Schaefer in combination with Gaines and/or Tormasov.
----	-----------------------------	--

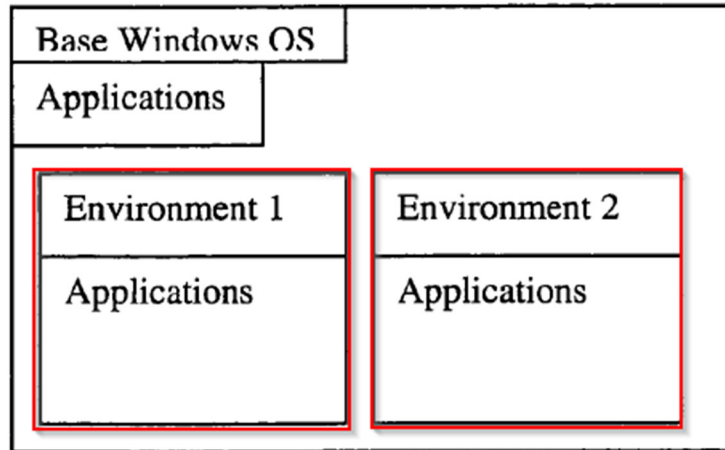
Section VII details the statutory grounds for unpatentability of each of the Challenged Claims. Additional evidence is provided in the accompanying Declaration of Dr. Mowry (Ex-1003) and other supporting exhibits. The sources of background information cited below are all dated prior to September 2003. Ex-1009; Ex-1010; Ex-1017; Ex-1018.

IV. OVERVIEW OF THE ASSERTED PRIOR ART

A. McMillan

Among its embodiments, McMillan discloses the use of secure containers stored in a central repository that can be accessed by multiple different servers with the same or different base operating systems.

McMillan discloses multiple separate virtual operating system (OS) environments that work with base OS. Ex-1005, 1:12-14. These virtual OS environments are each a “container” as described and claimed in the ’814 Patent. This is illustrated in Figure 1 of McMillan, which shows a server with a base OS as well as two containers (annotated in red):

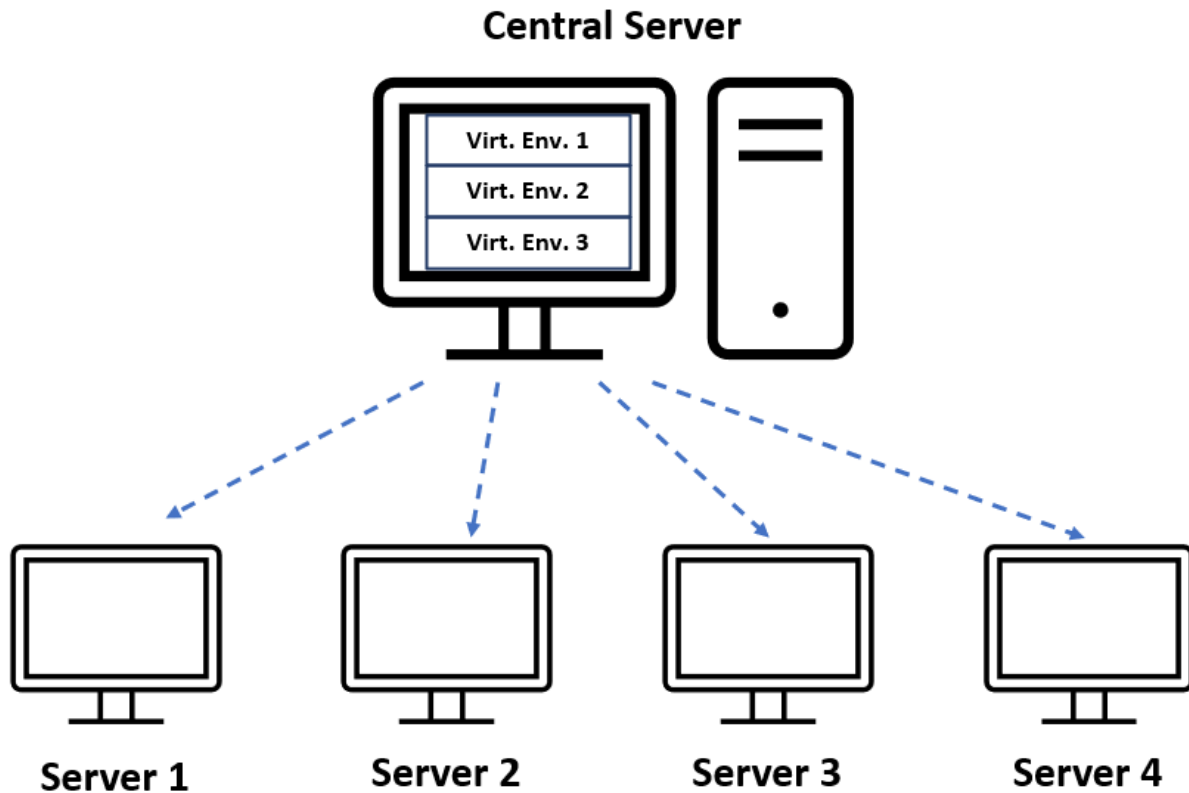


Ex-1005, Fig. 1 (annotated).

Each container includes one or more applications as well as additional files needed to run the applications, such as a copy of the base OS file system and registry. Ex-1005, 4:21-25. As a result, each environment runs independently of the others, and changes to one environment do not affect the other environments. *Id.*, 2:55-64. As explained in Section VII.A.1, McMillan’s environments are containers within the meaning of the ’814 Patent. Ex-1001, 2:23-25. Further, these containers are secure containers, because each container is insulated from the other containers and the base OS. *See* Ex-1005, 2:15-19, 2:57-60; Ex-1001, 2:43-48.

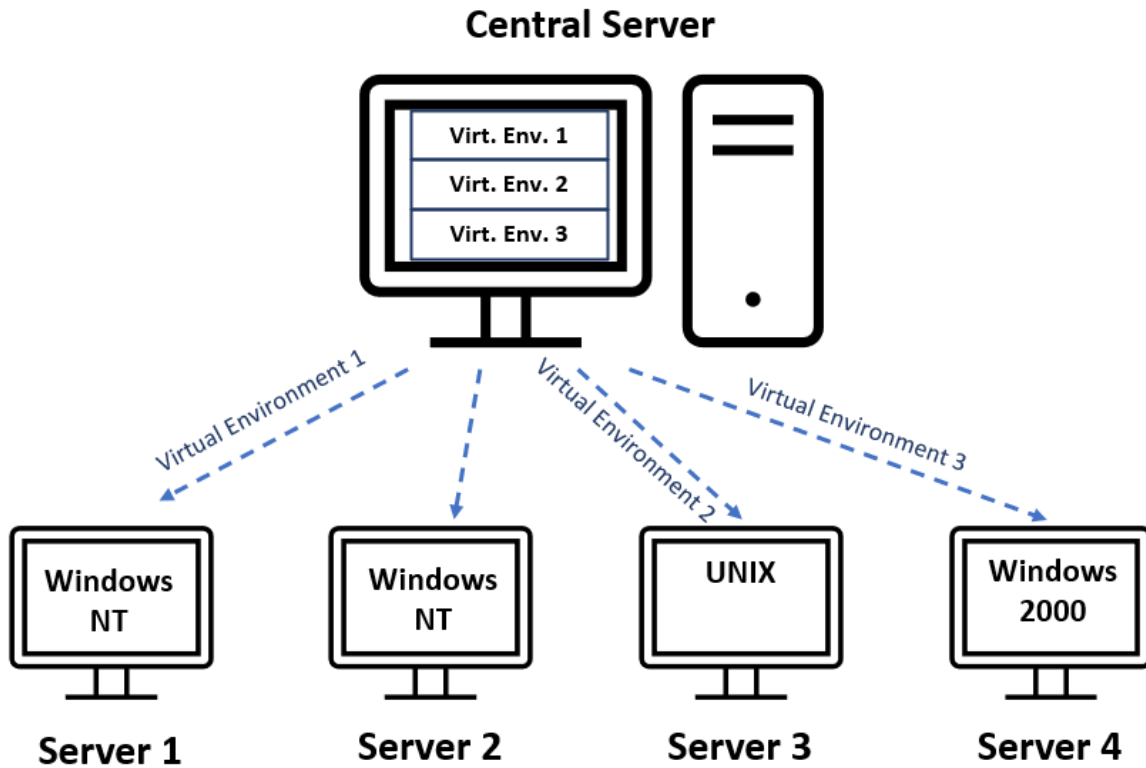
McMillan also discloses an embodiment in which its environments (containers) may be stored in a central location accessible to multiple computers, such as a central server. Ex-1005, 4:57-65 (containers may be stored “in a separate or central location to be used by multiple computers or the same computer at different times.”). Thus, as Dr. Mowry explains, McMillan discloses an architecture

where containers may be centrally located on a central server and accessible by multiple other servers:



Ex-1003, ¶49. Because McMillan describes but does not depict its multi-server embodiment, Dr. Mowry created the above schematic of McMillan's disclosure for illustrative purposes. Ex-1003, ¶49.

McMillan also explains that while its disclosed embodiments use a version of the Windows OS, which at the time included versions such as Windows NT and Windows 2000, McMillan is not limited to a Windows OS, but instead, may be used with other OSs, exactly as claimed in the '814 Patent. Ex-1005, 2:64-66. Thus, each container may be geared towards a particular operating system:



Ex-1003, ¶50. As shown above, a user of a server that desires to execute a McMillan container compatible with the base OS of that server can download that container from the central server and run the container locally. Ex-1005, 4:57-65; Ex-1003, ¶50.

B. Schaefer

Like McMillan, Schaefer teaches the use of software containers—referred to as “private contexts” in Schaefer—to facilitate application execution without requiring changes to an underlying OS.

As Schaefer describes, a private context (container) includes an application to be executed, as well as an OS abstraction and protection layer that is provided by a component Schaefer refers to as OS Guard. The OS Guard is interposed between one or more applications and the operating system. Ex-1008, [0013], [0063]. The abstraction and protection layer includes system files and “presents an environment to an application that appears to be an installation environment without performing an installation, whereby a ‘pseudo installation’ is created in which all of the settings are brought into a virtual environment at the time the application runs.” Ex-1008, [0007]; *see also id.*, [0014], [0027], [0040-41], [0058-60].

Tellingly, the European Patent Office (“EPO”) rejected the European counterpart of the ’814 Patent on the basis of Schaefer. PO was unable to overcome that rejection, and the application was ultimately deemed withdrawn due to abandonment. *See* Ex-1013 (EP FH), 217. During that prosecution, the applicant did not contest Schaefer’s disclosure of containers, but argued that Schaefer did not disclose, for example, a unique identity of the containers, a plurality of different servers with operating systems that differ, and that each container having a unique root file system. The EPO, however, correctly rejected those arguments, as discussed below. While cited on the face of the ’814 Patent, Schaefer was not subject to any substantive examination during original prosecution before the USPTO, alone or in

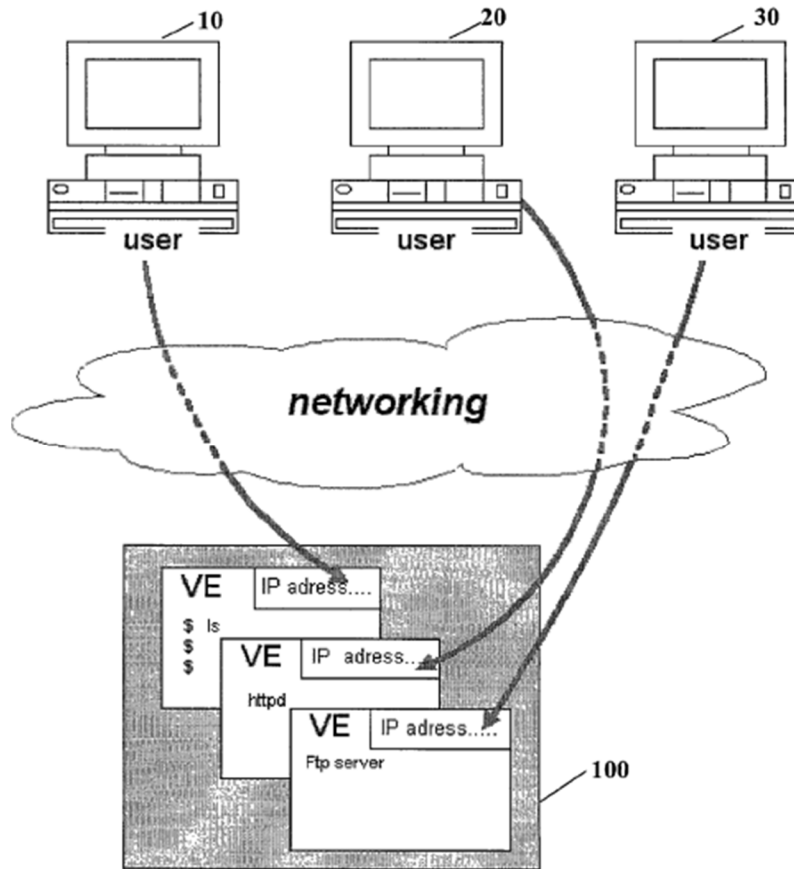
combination with the Gaines and/or Tormasov references with which Schaefer is combined.

C. Gaines

Gaines discloses a distributed executed environment comprising multiple servers running different OSs on which applications can run without concern for or knowledge of the underlying OS. For example, as illustrated in Gaines' Figure 3A, each host computer 101 (server) is coupled to a network and controlled by an OS, in which "each set of computer hardware 102 may differ from each other, and each operating system 103 may also differ from each other." Ex-1007, 12:60-62, Fig. 3A. Gaines "provides a virtual execution environment for which programs could be written that are capable of execution on different computer hardware and with different operating systems." Ex-1007, 3:66-4:2. Gaines allows for applications to be executed across different OSs through the use of a set of system files provided by a virtual application interface, such as a process control subsystem and virtual file subsystem, that effectively translates resources necessary for the application to run on the underlying operating system of the host computer, whatever it may be. Ex-1007, 4:23-39; *see also id.*, 6:36-8:32.

D. Tormasov

Tormasov discloses a networked environment with multiple user computers connected to a central server over a network:



Ex-1011, Fig. 1; *see also id.*, [0024]. The central server (100) stores a plurality of virtual environments (*i.e.*, containers) that each include an IP address for identification purposes, as illustrated in Figure 1. *Id.*, [0024]; Ex-1003, ¶62.

V. LEVEL OF ORDINARY SKILL IN THE ART

As of September 2003, a POSITA would have been an individual with a bachelor's degree in computer science or a similar degree and at least two years of experience in operating systems or computer systems software. This level of skill in the art is reflected by the references cited in this Petition, the state of the art, and the opinion of Professor Todd Mowry set forth in his Declaration. Ex-1003, ¶¶33–

36. In this Petition, reference to a POSITA refers to an individual with these qualifications.

VI. CLAIM CONSTRUCTION

Claim terms “shall be construed using the same claim construction standard that would be used to construe the claim in a civil action under 35 U.S.C. § 282(b).” 37 C.F.R. § 42.100(b) (2018); *Phillips v. AWH Corp.*, 415 F.3d 1303 (Fed. Cir. 2005) (en banc).

In the related, consolidated litigation *VirtaMove, Corp. v. Hewlett Packard Enterprise Co.*, 2:24-cv-00093-JRG (lead case) and *VirtaMove, Corp. v. International Business Machines Corp.*, 2:24-cv-00064-JRG (member case), the parties have agreed to the following constructions for the ’814 Patent:

Term	Claim(s)	Agreed Construction
Preambles	1	The preambles are limiting
Processor	1	Physical computer processor
Secure container / Secure container of application software	1, 2, 6, 9, 10, 31	“A container environment where each application set appears to have individual control of some critical system resources and/or where data within each application set is insulated from effects of other application sets,” wherein “each container for use on a server is mutually exclusive of the other containers, such that read/write files within a

		container cannot be shared with other containers
Within a container	1	Associated with a container
A kernel a set of associated local system files	1	A kernel and a set of associated local system files

Ex-1015, Appendix A. The parties also agreed that various terms and their antecedent refer to the same entity based on the plain claim language. *See id.*

PO has proposed that, except for the term “disparate computing environments,” all terms in the Challenged Claims be given their plain and ordinary meaning. *See id.*, Appendix C. PO has taken the position that “disparate computing environments” means “environments run by standalone computers.” *Id.*, Appendix C at 1. The ’814 Patent, however, defines “disparate computing environments” as “environments where computers are stand-alone or where there are plural computers and where they are unrelated,” and Petitioner proposes that construction. Ex-1001, 2:17-19, 2:20-22, 14:35-36.

Pursuant to 35 U.S.C. § 301(a)(2), fairness dictates that PO should be held to its claim construction positions taken in the related litigation. *See, e.g., Western Digital Corp. v. SPEX Techs., Inc.*, IPR2018-00084, Paper 14 at 12 (PTAB Apr. 25,

2018); *10X Genomics, Inc. v. Bio-Rad Laboratories, Inc.*, IPR2020-00089, Paper 8 at 12 (PTAB Apr. 27, 2020).

Petitioner also notes that the Asserted Claims are unpatentable using Petitioner’s proposed constructions and the various lexicographical definitions identified in the ’814 Patent at 2:17-54, and applies those definitions in its analysis as well.

VII. SPECIFIC GROUNDS FOR PETITION

A. Ground I: McMillan Alone or Alternatively In View of Gaines and/or Tormasov

As described in detail below, the Challenged Claims are obvious in view of McMillan, alone or in combination with Gaines and/or Tormasov. Ex-1003 ¶67.

1. Claim 1

a. 1 [Pre.a]

Limitation 1[Pre.a] is obvious. Ex-1003, ¶68.

First, McMillan discloses or renders obvious a plurality of servers. As indicated above, McMillan discloses in its Figure 1 embodiment a single server with a base OS housing two virtual OS environments. These virtual OS environments are the claimed “containers” in limitation [1a]. *See* Section VII.A.1.d. Petitioner hereby includes the corresponding page number of McMillan’s provisional application disclosure following the citations to McMillan (Ex-1005) in brackets []. McMillan also teaches storing virtual OS environments in a central location, such as a central

server, that can be accessed and run by multiple other computers. Ex-1005, 4:60-65 [6]. Further, each of these computers is a server because it provides a service to the users of a computer by executing an application downloaded from the central server as part of a container on behalf of the users. Ex-1003, ¶¶68–72; Ex-1005, 4:60-65 [6].

Second, McMillan discloses or renders obvious that these multiple computers (servers) may have operating systems that differ. McMillan is directed at the use of virtual OS environments (containers) with the goal of increasing system usability and versatility. Ex-1003, ¶¶73–77; Ex-1005, 2:64-67 [1], 4:30-32 [4], 4:60-65 [6]. While McMillan’s specific disclosed embodiments use a Windows OS, McMillan expressly teaches that its “approach is applicable to other operating systems through appropriate modification.” Ex-1005, 2:64-67 [1]. A POSITA would understand and find obvious that McMillan’s virtual OS environments (containers) may be used with multiple different operating systems to expand the flexibility, versatility, and usability of McMillan’s system. This would have the clear benefit of allowing the systems to be used with multiple different computers regardless of their OS, increasing the number of users who can use McMillan’s system. By not locking the system into a single OS, the versatility, flexibility, and usability of McMillan’s system is greatly increased. Ex-1003, ¶¶73–77.

A POSITA would also have been motivated to modify McMillan with Gaines' teaching of the use of different operating systems in order to improve system efficiency, flexibility, and versatility, particularly given McMillan's express disclosure that its system may be used with multiple different OSs. Ex-1003, ¶¶78. Gaines likewise teaches a system that provides secure virtual OS environments for program execution on host computers coupled to a network, where these computers may each have a different OS. *See* Ex-1007, 5:24-28, Fig. 1 (virtual OS environment within host computer connected to network), Fig. 3A (distributed execution environment). Gaines' teaching of multiple host computers, each with a different OS, increases the overall versatility, flexibility, and usability of the system because it increases the number of users/applications that can use Gaines' execution environment. Ex-1003, ¶¶78–79. A POSITA seeking to implement McMillan's system would be naturally motivated to use Gaines' multi-operating-system approach to attain these same universal goals for any well-designed computer system. This would be accomplished by having multiple different virtual OS environments geared towards different OSs stored in McMillan's system. Ex-1003, ¶¶78–80.

As Dr. Mowry explains, this common-sense modification of McMillan to have it function with multiple different OSs would be nothing more than the use of a known technique to improve similar systems in the same way. Ex-1003, ¶¶79–80.

It would also be a combination of prior art elements according to known methods to yield predictable results. The use of multiple OSs in a computer system was well known in the art and readily implementable by a POSITA. Ex-1003, ¶¶79–80. It would also be obvious to try as one of a finite number of possible configurations. Further, modifying McMillan’s system to use multiple OSs would involve routine and well-known computer programming techniques with no requirements for hardware modification. It would only require installation of different commercially available OSs on different computers, and programming each container to work with particular OSs that were commonly deployed in practice. Programming for such commonly deployed OSs would have been routine to a POSITA. *Id.* As such, a POSITA would have a reasonable expectation of success in implementing such a modification with predictable results. *Id.*, ¶80.

Third, McMillan discloses or renders obvious that its multiple computers (servers) operate in disparate computing environments, under both parties’ constructions. McMillan meets PO’s construction of “environments run by standalone computers,” because each of McMillan’s “multiple computers” is separate from the others and runs a virtual OS environment independently of the other computers. Ex-1005, 4:57-65 [6]; Ex-1003, ¶81. McMillan also meets Petitioner’s proposed construction, based on lexicography, of “[e]nvironments where computers are stand-alone or where there are plural computers and where they

are unrelated” (Ex-1001, 2:17-19), for this same reason, and because McMillan discloses the use of plural computers with no relationship among them. Ex-1005, 4:60-65 [6]; Ex-1003, ¶81.

Fourth, McMillan discloses or renders obvious that each of its multiple computers (servers) includes a processor and an OS including a kernel [and] a set of associated local system files compatible with the processor. First, a POSITA would understand and find obvious that a computer includes a physical computer processor—the presence of a processor is the key feature that makes a computer a computer. Ex-1003, ¶83; Ex-1009, 45 (“The ‘brain’ of the computer is the CPU.”), Figs. 1-5 (illustrating “CPU”, i.e., processor). A POSITA would also understand that McMillan’s multiple references to “running” an application and similar language indicate the presence of a physical processor, because processors run software such as applications. Ex-1003, ¶83. Second, McMillan expressly discloses that each of its computers has a base OS. For example, McMillan expressly teaches that applications can run in a virtual OS environment, *or* alternatively, use the base OS—thus McMillan discloses that each computer contains an OS (*i.e.*, the base OS). *See* Ex-1005, 3:4-5 (“all of the applications run *on a single OS desktop*.”) [1], 3:21-26 [2]. Third, a POSITA would understand and find obvious that any OS has a kernel, because a kernel controls the interaction between a computer’s hardware and the software executing on a computer. Ex-1003, ¶¶84–85; Ex-1009, 463, Fig. 11–7.

Fourth, McMillan discloses that a base OS has associated system files, which were well-known to a POSITA and present in all OSs. *See* Ex-1003, ¶86; Ex-1009, 62. For example, McMillan explicitly teaches that each computer (server) includes a base OS file system and registry, as further illustrated in Figure 2:

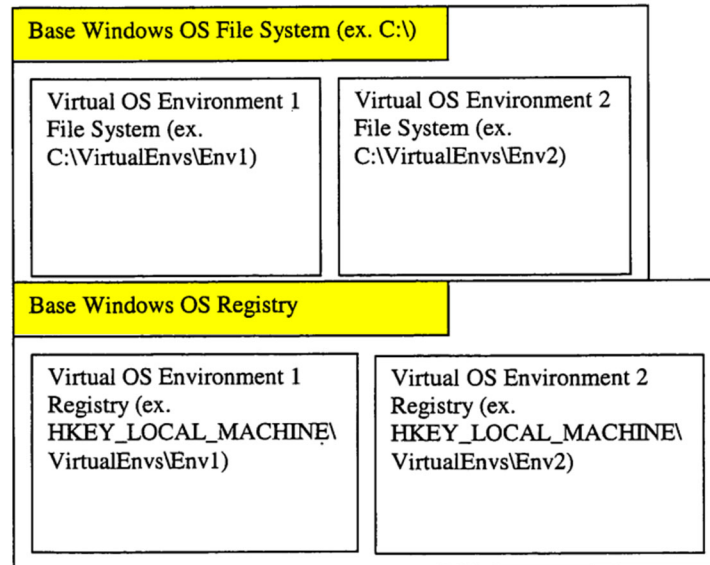


FIGURE 2

Ex-1005, Fig. 2 (annotated) [2].

An application accesses a file system and registry for files, services, and/or other information required for the application to run effectively. Ex-1005, 3:17-30 [2], 4:21-32 [4]; *see also id.*, 4:15-16 [4], 4:21-25 [4], 5:20-23 [6]; Ex-1003, ¶87. A POSITA would understand and find it obvious that the base OS file system and registry include “system files provided within an operating system and which are available to applications as shared libraries and configuration files.” Ex-1001, 2:52-54; *see also id.*, 5:20-26. Further, a POSITA would understand McMillan’s system

files to include dynamically-linked libraries (“DLLs”), or shared libraries, because DLLs provide applications with services to access portions of any given file system. *See* Ex-1005, 3:1-10 [1], 3:17-30 [2], 4:33-36 [4]; Ex-1003, ¶¶88–89. A POSITA would understand that the base OS file system and registry provides system files to an application, and that those system files would be wrapped and made available to an application as a DLL, or shared library. *See* Ex-1005, 3:17-48 [2-3], 4:33-36 [4]; Ex-1003, ¶90. McMillan expressly discloses the use of system calls and APIs, which utilize DLLs, to access the file system and registry, which a POSITA would recognize as the mechanism by which an application would access system files, including services, in the file system. *See* Ex-1005, 3:17-47 [2-3], 4:21-5:8 [4-6].

b. 1 [Pre.b]

Limitation 1[Pre.b] is obvious. Ex-1003, ¶91.

First, McMillan discloses or renders obvious a method of providing at least some of the servers in the system with secure, executable applications related to a service. Ex-1003, ¶¶92–93. As previously discussed, McMillan teaches the ability to “save, store, and load entire virtual OS environments,” such as “in a separate or central location to be used by multiple computers.” Ex-1005, 4:57-65 [6]. These virtual OS environments (containers) include applications, such as NotePad and WordPad, which relate to a service such as word processing. *See id.*, 4:25-30 [4]. McMillan discloses that these applications are secure because the applications within

virtual OS environments “run independently of each other” and the base OS. *See* Ex-1005, 2:15-19 [1], 2:60-62 [1].

Second, McMillan discloses that these applications are executed in a secure environment. McMillan’s virtual OS environments (containers) are secure because they are independent and isolated environments that cannot affect the base OS or any other environment (*i.e.*, secure environments). Ex-1005, 2:57-60 [1]. McMillan explains that “*[t]he file system and registry information for each environment is independent of the base OS and other environments*” and that the environments’ applications “*will run independently of each other.*” Ex-1005, 2:60-64 [1]; *see also id.*, 3:11–16 [1–2], Cl. 4 [Cl. 5] [8]. This is consistent with how the ’814 Patent teaches applications executing in a secure environment, which “is made possible by associating applications with secure containers” such that “[a]pplications are segregated *and execute with their own copies of files* and with a unique identity.” Ex-1001, 7:4-10; Ex-1003, ¶¶94–95.

c. 1 [Pre.c]

Limitation 1[Pre.c] is obvious. Ex-1003, ¶96.

First, McMillan discloses or renders obvious that “the applications each include an object executable ... for performing a task related to the service.” Ex-1003, ¶97. For example, McMillan discloses that applications that run in the Virtual OS environments include applications to be executed “such as Notepad™ text editor,

Wordpad™ word processor, or Internet Explorer® internet browser, all from Microsoft Corporation” which “will be executed from the virtual OS environment copy of that application.” Ex-1005, 4:25-30 [4], *see also id.*, 4:30-32 [4]. These applications include an EXE, or an executable file, which a POSITA would understand includes an object executable. Ex-1005, 3:59-61 [3]; Ex-1003, ¶97. A POSITA would understand that McMillan’s EXE file is an object executable, because it is written in binary format to be executed by the processor. *See* Ex-1003, ¶97; Ex-1009, 291, 498; Ex-1010, 5. A POSITA would further understand that these applications are for performing a task related to the service. For example, a service provided by Notepad™ text editor or Wordpad™ word processor can include editing a document, and specific tasks related to the service of editing the document can include typing words, font formatting, text manipulation, etc. Ex-1003, ¶97.

Second, McMillan discloses or renders obvious that the applications run in the virtual OS environments can be executed “by at least some of the different operating systems.” Ex-1003, ¶98. For example, consider a system of 5 servers where two servers are running Unix, two servers are running Windows NT, and another is running Windows 2000. These OSs were all commercially available and well-known when McMillan’s provisional was filed in November 18, 2002. Ex-1003, ¶98. An application written for Windows NT can be executed by one server running Windows NT, or both servers running Windows NT, which are “at least

some of the different operating systems.” McMillan discloses that its containers can be saved, stored, and loaded, and that the containers can be stored in a central location to be downloaded and used by multiple computers. *See* Ex-1005, 4:57-65 [6]. Thus, a POSITA would understand that McMillan discloses the ability that two different computers (e.g. both running different instances of Windows NT) can download a virtual OS environment from a central location (memory) to execute the applications as disclosed in McMillan. Ex-1003, ¶98; *see also* Ex-1005, 4:21-32 [4], Figs. 1-2 [1-2].

Further, McMillan discusses in depth the use of the OS Application Programming Interface (API), which a POSITA would understand to be the Windows Win32 API given that McMillan’s disclosed embodiments are all based on Windows. *See* Ex-1005, 3:17-48 [2-3], 4:33-56 [4-5], Figs. 3, 5, 6 [3-5]; Ex-1003, ¶99. A POSITA would understand that computers using both Windows NT and Windows 2000 would use the Win32 API, which would allow the same application to run on both versions of Windows. Ex-1003, ¶99; Ex-1009, 456-458. Indeed, as Dr. Mowry explains, it was well known to a POSITA that the Win32 API “allows programs to run on almost all versions Windows,” including Windows NT and Windows 2000:

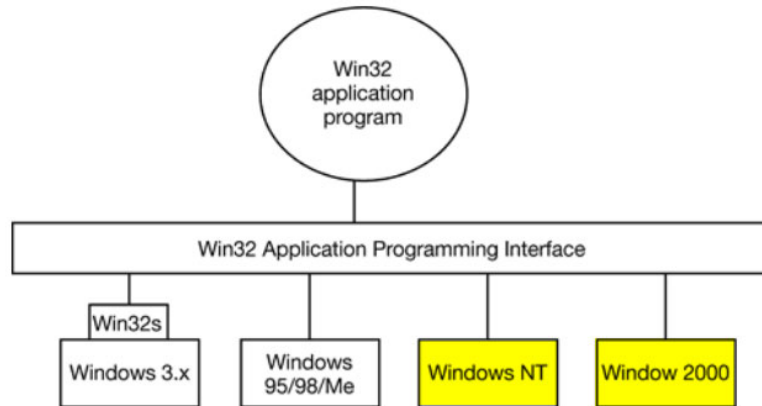


Figure 11-4. The Win32 API allows programs to run on almost all versions of Windows.

Ex-1009, 456, Fig. 11-4 (annotated). Given that Windows OSs are backwards compatible, a POSITA would understand and find obvious that an application formatted for Windows NT can be run on Windows NT and Windows 2000.

d. 1[a]

Limitation 1[a] is obvious. Ex-1003 ¶100.

First, McMillan discloses “storing in memory accessible to at least some of the servers a plurality of secure containers of application software.” Ex-1003, ¶101. Specifically, McMillan discloses storing virtual OS environments in a central location accessible to multiple computers (servers), as discussed in section VII.A.1.a. *See supra* VII.A.1.a. A POSITA would understand that because these virtual OS environments are software, these environments are stored in memory in the central location such as a hard disk or other storage. Ex-1003, ¶100. And, as previously discussed, these virtual OS environments are a plurality of secure containers of application software. *See supra* Claim 1 [Pre.b]; Ex-1003, ¶100.

Second, McMillan discloses “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.” Ex-1003, ¶102. Specifically, McMillan teaches that the virtual OS environments (containers) contain a copy of a subset of the base OS system files, and explains that “any applications that are run from under the virtual OS environment file system (ex: C:\VirtualEnvs\Env1) are always redirected into that virtual OS environment.” Ex-1005, 4:4-16 [4], Fig. 5; *see also id.*, 4:21-30 [4].

Third, McMillan discloses or renders obvious that the secure application containers are “for use with a local kernel residing permanently on one of the servers.” Ex-1003, ¶103. McMillan teaches that “[t]he applications running within the virtual OS environments still share the base OS attributes such as **networking information**, user login rights, services, **hardware information**, and the Windows clipboard.” Ex-1005, 3:1-4 [1]; *see also id.*, Cl. 2 [Cl. 3] [8]. A POSITA would understand and find obvious that networking information and hardware information are types of information within the kernel. Ex-1003, ¶¶103–104; Ex-1012, [0005]; Ex-1010, 1. Additionally, McMillan consistently references the use of APIs, which in McMillan’s primarily Windows-based system, a POSITA would understand to be the Win32 API. Ex-1003, ¶104. A POSITA would understand that part of Win32 API is run in kernel mode and interacts with the kernel. *See* Ex-1003, ¶104; Ex-1010, 2 (illustrating “Win32” in “kernel mode”).

e. 1[b]

Limitation 1[b] is obvious. Ex-1003, ¶105.

As previously discussed, McMillan teaches that the virtual OS environments (*i.e.*, secure containers) can be used for different OSs by multiple computers. A POSITA would have understood and found obvious that the set of associated system files in McMillan, which are the copies of the base OS file system and registry within the virtual OS environment, would be compatible with a local kernel of at least some of the plurality of different OSs.

For example, consider again a configuration of five servers connected to a central server where two are running Unix, two are running Windows NT, and another is running Windows 2000. A container designed for operation with a base Windows NT OS, the system files would be compatible with the local kernel of the computers running Windows NT as a base OS. These two instances of Windows NT are “at least some of the plurality of different operating systems.” Ex-1003, ¶¶106–107. Additionally, the set of associated system files would also be compatible with the local kernel of the computers running Windows 2000 as a base OS, as both Windows NT and Windows 2000 utilize Win32 API. *See* Ex-1005, 3:34-48 (virtual environments use injected DLL to modify parameters and call base OS API) [2-3]; Ex-1003, ¶107; Ex-1009, 456, Fig. 11-4 (“The Win32 API allows programs to run

on almost all versions of Windows.”) (illustrating Windows NT and Windows 2000); *see also* Ex-1005, Figs. 5 [4], 6B [5].

Furthermore, Gaines expressly discloses a “virtual operating system is capable of executing programs in a standardized programming language...to provide the remote user ***with the ability to run programs that are host-independent.***” Ex-1007, 4:13-18; *see also id.*, 6:22-35. Indeed, Gaines teaches a plurality of computers with different operating systems, and that “[t]he host virtual operating system ***may be programmed for any of these hardware and operating system software.***” Ex-1007, 5:1-16, 3:66-4:2. A POSITA would have understood and found obvious that it could incorporate Gaines’ teaching of its “virtual application interface,” “resource filters,” and “language engine” with McMillan’s virtual OS environments, allowing the same container to be used on different OSs such as Windows and Unix and thereby increasing the usability of McMillan’s system. Ex-1003, ¶¶108–109. This would have been an application of a known technology to McMillan with predictable results. Ex-1003, ¶110. This also would have been obvious to try as one of a finite number of implementations. A POSITA would also have had a reasonable expectation of success, given that implementation would involve only software programming and no hardware modifications. Ex-1003, ¶110.

f. 1[c]

Limitation 1[c] is obvious. Ex-1003, ¶111.

First, McMillan teaches that the virtual OS environments (containers) exclude a kernel. McMillan explains that “[t]he applications running within the virtual OS environments still share the base OS attributes such as **networking information**, user login rights, services, **hardware information**, and the Windows clipboard.” Ex-1005, 3:1-4 [1]; *see also id.*, Cl. 2 [Cl. 3] [8]. A POSITA would understand and find it obvious that networking information and hardware information are types of information provided by the kernel. Ex-1003, ¶113; Ex-1012, [0005]; Ex-1010, 1. If the containers contained their own kernel, they would not need to rely on the base OS for this information. Additionally, McMillan consistently references the use of APIs, which in McMillan’s primarily Windows-based system, a POSITA would understand to be the Win32 API. Ex-1003, ¶113. A POSITA would understand that part of Win32 API is run in kernel mode and located outside of the containers. *See* Ex-1003, ¶113; Ex-1010, 2 (illustrating “Win32” in “kernel mode”).

Second, McMillan’s virtual environments (containers) utilize copies of certain base OS system files and registries that are associated with a container in running the application software in place of the associated local system files that remain resident on the server, because they use the container files and not the corresponding kernel files. Ex-1005, 4:21-25 [4], Fig. 2 (illustrating Base OS File System and Registry) [1-2], Fig. 7 [6]; *see also supra* Claim 1[b]; Ex-1001, 10:18-34, 10:41–46.

g. 1[d]

Limitation 1[d] is obvious. Ex-1003, ¶115. As previously discussed for limitation 1[c], McMillan teaches that each virtual OS environment (containers of application software) utilizes *copies* of certain base OS system files (*i.e.*, associated local system files that remain resident on the server). *See supra* Claim 1[c]; Ex-1003, ¶116.

h. 1[e]

Limitation 1[e] is obvious. Ex-1003, ¶117.

First, McMillan discloses or renders obvious “wherein the application software cannot be shared between the plurality of secure containers of application software.” Ex-1003, ¶118. As previously discussed, McMillan teaches that each virtual OS environment is an independent environment such that “[a] change made in one environment does not affect the main OS or any other environment.” Ex-1005, 2:57-58 [1]; *see also id.*, 2:59-60 (“[E]ach virtual OS environment appears to be an independent OS for the applications running within it.”) [1], 2:62-64 (“Each of the environments can contain a group of installed applications that will run *independently of each other.*”) [1]; Cls. 3 [Cl 4] [8], 4 [Cl. 5] [8], 6 [Cl. 7] [8]. McMillan also teaches that an application is specific to a virtual environment, and not shared between environments. This is shown, for example, in McMillan Figure 1, where each container has its own applications. Ex-1005, Fig. 1 [1], 2:62-64 [1];

Ex-1003, ¶118. Thus, a POSITA would understand from McMillan’s teachings that the application software running within each virtual OS environment cannot be shared with other virtual OS environments, as the environments are all independent, and as discussed below, have their own file system and registry information. Ex-1003, ¶118.

Second, McMillan discloses or renders obvious “wherein each of the containers has a unique root file system that is different from an operating system’s root file system.” Ex-1003, ¶119. As previously discussed, McMillan teaches that each virtual OS environment (container) utilizes a copy of base OS file system and registry—which include separate directory locations, and thus unique root file systems. *See supra* Claim 1[c]; Ex-1005, 4:21-25 [4], Fig. 5 [4]; Ex-1003, ¶119. Indeed, McMillan explicitly teaches that “[t]he file system and registry information for each environment is independent of the base OS and other environments” and in independent directories. Ex-1005, 2:60-62 [1]; *see also id.*, 3:11-13 [1], 3:49-65 [3], Cl. 1 [Cl.1] [8], Figs. 2 [1-2], 4 [3], 5 [4], 7 [6].

2. Claim 2

Claim 1 is obvious. Claim 2 is also obvious. Ex-1003, ¶120. As previously discussed, McMillan discloses that the applications that run in its Virtual OS environments (containers) include applications to be executed “such as Notepad™ text editor, Wordpad™ word processor, or Internet Explorer® internet browser, all

from Microsoft Corporation” which “will be executed from the virtual OS environment copy of that application.” Ex-1005, 4:25-32 [4]. McMillan explains that each application has an EXE, or executable, that when run, starts the application. Ex-1005, 3:59-61 [3]; Ex-1003, ¶121. A POSITA would understand and find obvious that “an application EXE” is an execution file associated with starting the given application. Ex-1003, ¶121; Ex-1009, 498; *see also id.*, 291; Ex-1010, 5 (“Each executable image (.EXE)...”).

3. Claim 6

Claim 2 is obvious. *See supra* Claim 2. Claim 6 is also obvious. Ex-1003, ¶122.

First, McMillan discloses that each virtual OS environment (container) is assigned a unique identity. As shown in McMillan’s Figures 1 and Figure 2 below, and further explained by Dr. Mowry, each virtual OS environment (container) includes a unique name (*i.e.*, Virtual OS Environment 1 (Env1) or Virtual OS Environment 2 (Env2)):

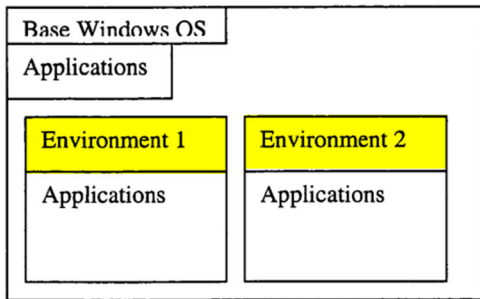


FIGURE 1

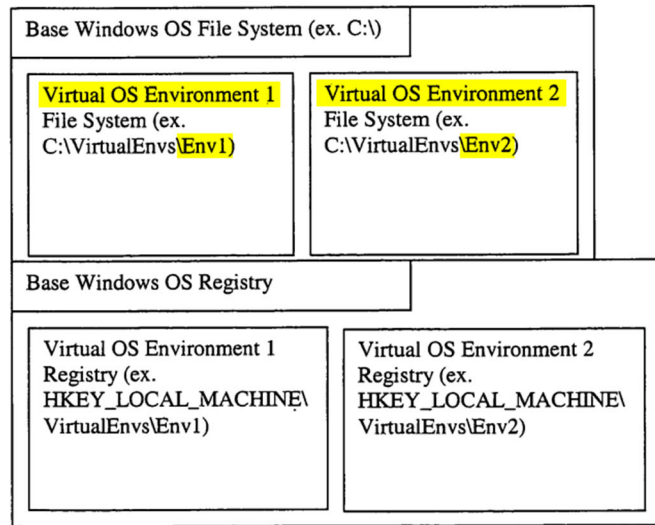


FIGURE 2

Ex-1005, Figs. 1-2 (annotated) [1-2]; *see also id.*, 4:4-7 (“applications that are run from under the virtual OS environment file system (ex: C:\VirtualEnvs\Env1) are always redirected into that virtual OS environment.”) [4]; Ex-1003, ¶123; *see also* Ex-1005, Figs. 4-6 [3-5]. This is consistent with McMillan’s teaching that its containers are specific. For example, McMillan explains that “[t]he base OS contains a list of directories where applications should be redirected *to specific virtual OS environments*” and “[a]ny application EXE that is executed in this directory or optionally a child of this directory *will run under a specific virtual OS environment.*” Ex-1005, 3:57-61 [3]. Gaines also teaches that a unique identity is assigned to particular copies of the virtual OS. *See* Ex-1007, 11:19-23.

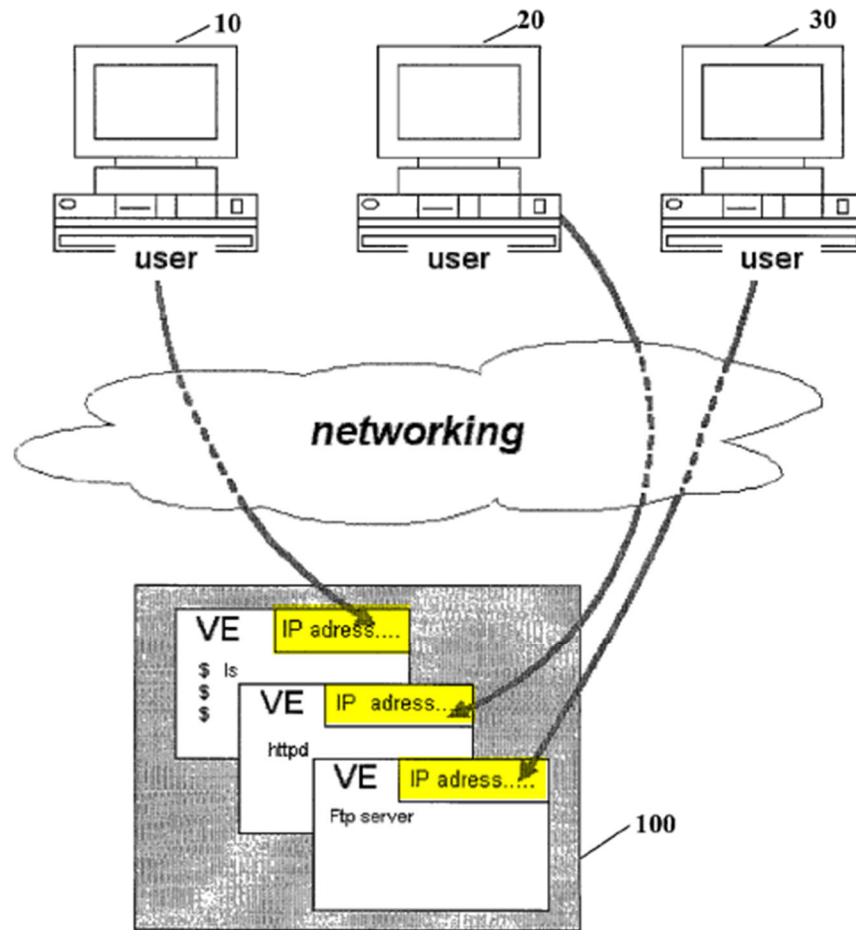
Second, McMillan discloses or renders obvious that each unique identity further includes at least host name, MAC address, and/or IP address. For example,

McMillan discloses that its environments are in a specific virtual directory, such as “C:\VirtualEnvs” or on a specific machine, such as “HKEY_LOCAL_MACHINE\VirtualEnvs.” These are names that identify where the containers are hosted, and thus are host names. In addition, a POSITA would understand that in Windows OSs, a Windows parameter known as “hostname” is found in the Windows registry and is used to identify the computer (*i.e.*, server) hosting any given container. *See* Ex-1003, ¶125; Ex-1009, 459-461 (“The first key [in Windows 2000 Registry], HKEY_LOCAL_MACHINE, is probably the most important as it contains all the information about the local system [*i.e.*, computer].”). Thus, a POSITA would understand that each of McMillan’s uniquely identifiable virtual environments would have their own hostname parameter under their “VirtualEnvs” directory to identify the container.

Additionally, as previously discussed, McMillan discloses storing the virtual OS environments (containers) in a central location (memory) that can be accessed by multiple computers. *See* Ex-1005, 4:57-65 [6]; Ex-1003, ¶126. This requires a network configuration, as Dr. Mowry explains. Ex-1003, ¶126. In a network configuration, a POSITA would have found it obvious that the unique identity needs to include some way(s) for a computer seeking access to the container to locate and communicate with the central server on which is it stored. For example, a POSITA would have found it obvious to use a network to connect McMillan’s multiple

computers (servers) to a central location, such as the network taught by Gaines. In any such network, the machines on the network would have addresses. Thus, any container on the central server would have an associated network address. One common network at the time was an IP network, which uses IP addresses to identify devices and other resources on the network. It would have been obvious to a POSITA to use this well-known network type in McMillan so that uniquely identifiable containers could be located on the network and downloaded to a computer seeking to use the container. *See* Ex-1003, ¶126; Ex-1009, 420. Indeed, Gaines expressly references use of protocols (Telnet and FTP) that use IP addresses. Ex-1007, 6:8-21; Ex-1003, ¶126.

This would also be obvious in view Tormasov, which discloses virtual environments in an IP network that have an associated IP address:



Ex-1011, Fig. 1 (annotated), [0024] (“each virtual computing environment is the actual remote computer *with the network address*”). As Dr. Mowry explains, a POSITA would have found it obvious to apply Tormasov’s teachings of an IP network and associating an IP address with each of McMillan’s uniquely identifiable containers so that they can be located and accessed on the IP network. Ex-1003, ¶127. Such a combination would amount to use of a known technique to improve similar devices in the same way, and also application of a known technique to a known system ready for improvement to yield predictable results. Ex-1003, ¶128.

It is also consistent with the general knowledge of a POSITA, as discussed above. Ex-1003, ¶128.

Additionally, a POSITA would have understood that MAC addresses were well-known address types used to identify particular devices when connected within a local network and facilitate communication amongst these devices. Ex-1003, ¶129. MAC addresses are used in many networking technologies, including Ethernet, Wi-Fi, and Bluetooth. *Id.*; *see also* Ex-1019, 1:6-2:11. As Dr. Mowry explains, a POSITA would have found it obvious and beneficial to use an associated MAC address to locate a container on the network, for the same reasons as it would an IP address. In fact, IP packets typically contain both an IP address and MAC address of the destination device. Ex-1003, ¶129. Thus, a POSITA would have found it obvious to include both an IP and MAC address with the unique identity so that the containers can be located and accessed by the other servers on the network, for the same reasons discussed above. Ex-1003, ¶129.

4. Claim 9

Claim 2 is obvious. *See supra* Claim 2. Claim 9 is also obvious. Ex-1003, ¶130.

As previously discussed, McMillan teaches that the virtual OS applications “still share the base OS attributes such as networking information, user login rights, services, hardware information, and the Windows clipboard.” Ex-1005, 3:1-4 [1].

As Dr. Mowry explains, the most basic of networking information includes network bandwidth, which tells a server and an application running on that server whether the application is able to communicate over the network. Additionally, as discussed above, each of McMillan's containers includes a file system and registry; this information provides disk allocation information. Ex-1003, ¶¶131–132.

Additionally, as previously discussed, McMillan teaches that its virtual OS environments can be loaded onto multiple computers from a central location. As Dr. Mowry explains, a POSITA would have found it obvious to associate a portion of CPU memory, network bandwidth, and/or disk allocation when loading the virtual OS environments on a given computer system prior to application execution, otherwise the application could not run on a given system. Ex-1003, ¶133. Gaines similarly teaches constraints on resources to be made available to given applications. *See* Ex-1007, 13:52-57; Ex-1003, ¶134.

5. Claim 10

Claim 2 is obvious. *See supra* Claim 2. Claim 10 is also obvious. Ex-1003, ¶135. McMillan teaches that applications executing within each virtual OS environment (container) are independent and without access to the underlying base OS system files or system files in other virtual OS environments. Ex-1003, ¶136. For example, McMillan teaches “[t]he file system and registry information for each environment is independent of the base OS and other environments.” Ex-1005, 2:60-

62 [1]; *see also id.*, Cls. 3 [Cl. 4] [8], 4 [Cl. 5] [8], 6 [Cl. 7] [8]. Thus, each application has no access to the base OS or other containers when executing. Further, McMillan explains that applications in a virtual OS environment access its own virtual OS file system or registry rather than the base OS file system and registry. Ex-1005, 3:11-16 [1], 4:42-45 [5]; Ex-1003, ¶137. McMillan also explains that an application running under a specific virtual OS environment “can only make changes to the virtual OS,” “can only place files into a location that is under the virtual OS environment itself,” and “cannot change any file system or registry information in the base OS file system.” Ex-1005, 3:59-4:16 [3-4]; *see also id.*, Figs. 1-2 [1-2].

B. Ground II: Schaefer Alone or in View of Gaines and/or Tormasov

As described in detail below, the Challenged Claims are obvious in view of Schaefer alone or in combination with Gaines and/or Tormasov. Ex-1003 ¶138.

1. Claim 1

a. 1 [Pre.a]

Limitation 1[Pre.a] is obvious. Ex-1003, ¶139.

First, “a system having a plurality of servers with operating systems that differ” is obvious.

For example, Schaefer discloses a networked software environment with at least one client computer that includes one or more applications, an OS abstraction and protection layer provided by an “OS Guard” component, and an OS. Ex-1008,

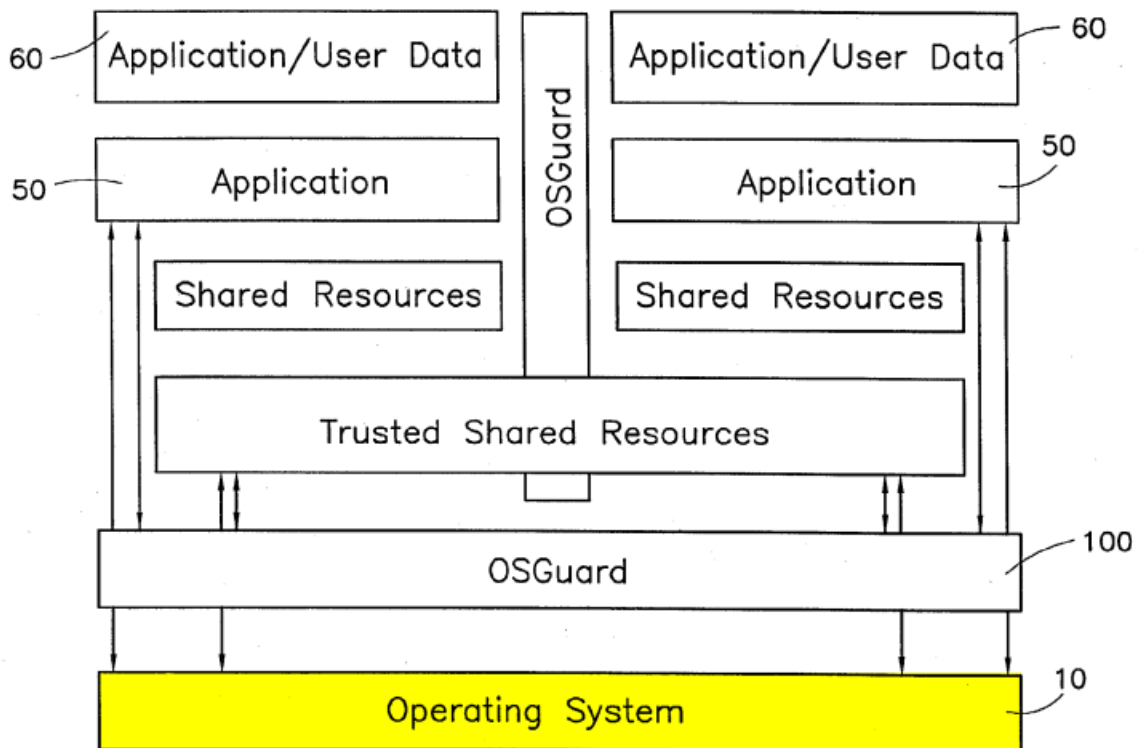
[0013], [0019], [0042], Figs. 1-2, 4. This computer is a server because it provides a service to the users of a computer by executing an application on behalf of the users. Ex-1003, ¶140. Additionally, because Schaefer is expressly designed to work in a networked environment, a POSITA would understand and find obvious that multiple of Schaefer's client computers (servers) may be deployed on Schaefer's network. Schaefer expressly teaches "[t]he present invention is particularly useful for applications delivered over a network..." Ex-1008, [0042]; Ex-1003, ¶142. Moreover, Schaefer expressly teaches its OS Guard—which is "a layer between a running application and the operating system of a target computer or client computer that provides a virtual environment in which an application may run"—is applicable to computers of differing OSs, and thus multiple computers. *See, e.g.,* Ex-1008, [0005] (discussing Windows embodiment), [0026] (discussing Windows, Macintosh, and UNIX), [0032] (discussing Windows, Macintosh, and UNIX), [0013], [0032]; Ex-1003, ¶143. Thus, it would at least have been obvious to deploy multiple of Schaefer's client computers (servers) in a networked configuration where different OSs are used. Indeed, the EPO expressly found this to be the case. Ex-1013 (EP FH), 169. The use of networked computers with different OSs was a well-known and common deployment in practice, and would increase the usability and versatility of Schaefer's networked environment, because it would not limit its use to a single OS. Ex-1003, ¶144.

The use of multiple computers and OSs would also be obvious in view of Gaines. As discussed above in Ground I, Gaines teaches a networked execution environment where different computers (servers) have different OSs and can execute applications on different virtual environments. This use of different OSs increases the usability, flexibility, and versatility of Gaines' system, because Gaines' system is not locked into any particular OS. As Dr. Mowry explains, a POSITA would be motivated to use multiple OSs with Schaefer to achieve these same benefits for all of the same reasons discussed in Section VII.A.1.a regarding McMillan, which is incorporated by reference as if fully set forth herein. Ex-1003, ¶¶145–146.

Second, in a networked deployment of Schaefer's client computers (servers) with different OSs, those computers (servers) would operate in disparate computing environments under both parties' constructions. They meet PO's construction because each of Schaefer's computers is separate from one another. Ex-1001, 2:17-19; Ex-1003, ¶147. They also meet Petitioner's proposed construction for this same reason, and because Schaefer's computers would be separate standalone computers connected to a network, with no relationship among them. Ex-1003, ¶147.

Third, the use of a processor and an operating system including a kernel [and] a set of associated local system files compatible with the processor is also obvious. First, as discussed in Section VII.a.1.a regarding McMillan, a POSITA would understand and find obvious that a computer includes a physical computer

processor—the presence of a processor is the key feature that makes a computer a computer. *See supra* VII.a.1.a; Ex-1003, ¶¶148–149. That section is incorporated by reference as if fully set forth herein. Second, as disclosed by Schaefer, each of its computers contains an OS, including as illustrated in Figure 1 below. *See* Ex-1008, [0012] (“Disposed between the application 50 and **operating system 10** ...”), [0063] (“the operating system guard 100 serves as an abstraction layer ... between the application software 50 and **the actual operating system 10 of the client computer.**”). A POSITA would understand and find obvious that an OS requires a processor in order to run. Ex-1003, ¶150.



Ex-1008, Fig. 1 (annotated). Third, a POSITA would understand and find obvious that any OS has a kernel, where a kernel controls the interaction between a computer's hardware and the software executing on a computer. Ex-1003, ¶151; Ex-1009, 463, Fig. 11–7. Fourth, Schaefer discloses that a base OS has associated system files, which were well-known to a POSITA and present in all OSs. *See* Ex-1003, ¶152; Ex-1009, 62. For example, and as the PO conceded to the EPO during prosecution, Schaefer discloses that the virtual environment contains duplicates, or copies, of the system files in the underlying OS of the computer. *See* Ex-1013 (EP FH), 150 (OS Guard “provided, for each application, more or less duplicates of the *system files of the operating system*” and “each application sees its own modified system files.”). Further, a POSITA would understand that Schaefer's system files include dynamically-linked libraries (“DLLs”), or shared libraries, as expressly discussed in Schaefer. *See, e.g.*, Ex-1008, Abstract (discussing DLLs and shared libraries), [0013] (same), [0032] (same); Ex-1003, 153–154.

b. 1 [Pre.b]

Limitation 1[Pre.b] is obvious. Ex-1003, ¶155.

First, Schaefer teaches an OS Guard which “defines a layer between a *running application* and the operating system of a target computer or client computer that *provides a virtual environment in which an application may run.*” Ex-1008, [0013]; *see also id.*, [0019]. Schaefer explicitly describes examples of the

applications that can run in the virtual environments such as Adobe Photoshop, Microsoft Office, and Microsoft Word. *See id.*, [0014], [0020]; Ex-1003, ¶¶156–157. A POSITA would understand that these applications can be related to multiple services, such as an advertising service or banking service as explicitly taught by Gaines. *See* Ex-1007, 11:47-52; Ex-1003, ¶157. These applications are “secure” because Schaefer’s OS Guard “prevents information that may exist on the client/users machine from interfering with or modifying the behavior of an application” and by allowing “each application ... to run in a private context within the system....” Ex-1008, [0015], [0019].

Second, Schaefer’s virtual environments provide secure environments for each application to execute. For example, as illustrated in Figure 2, two separate applications or different versions of the same application (52, 54) execute independently, and utilize different system files unique to them, thus providing secure environments, like the ’814 Patent. *See* Ex-1008, [0019]; *see also id.*, [0040], Fig. 4 (illustrating virtual environment manager with “files” and “configuration”); Ex-1001, 7:4-10; Ex-1003, ¶158. An application may not execute in the private context other than the one to which it is assigned. *See* Ex-1008, [0012], [0019-0020], [0040], Figs. 1-2; Ex-1003, ¶159.

c. 1 [Pre.c]

Limitation 1[Pre.c] is obvious. Ex-1003, ¶160.

First, Schaefer discloses or renders obvious that “the applications each include an object executable ... for performing a task related to the service.” Ex-1003, ¶¶161–162. As previously discussed, Schaefer explicitly describes examples of the applications that can run in the virtual environments, such as Adobe Photoshop, Microsoft Office, and Microsoft Word. *See* Ex-1008, [0014], [0020]. Schaefer also explicitly teaches that the virtual environments can provide private contexts for a given application to run. *See id.*, [0019], Figs. 1, 2; Ex-1003, ¶163. A POSITA would understand that, for example, Microsoft Excel can be used for performing a task (updating the data in a row) for a given service, such as an advertising service or banking service as explicitly taught by Gaines. *See* Ex-1007, 11:47-52; Ex-1003, ¶164. A POSITA would understand that the applications running in a given virtual environment would include an object executable, as Schaefer explicitly teaches that the virtual environments are provided for applications to run, and a running application requires executable object code. *See* Ex-1008, [0004], [0013-0014], [0019]; Ex-1003, ¶164; Ex-1009, 291, 489. This is consistent with the EPO’s findings, which PO did not dispute during prosecution. *See, e.g.*, Ex-1013 (EP FH), 188.

Second, it is obvious that the applications run in the virtual environments can be executed “by at least some of the different operating systems.” *See supra* Claim 1[Pre.a]; Ex-1003, ¶165. For example, consider a networked deployment of 5

Schaefer client computers (servers) in which two computers are running Unix, two computers are running Windows NT, and one computer is running Windows 2000. As described above regarding McMillan, any application written for one of these Windows OSs would be executable on a different instance of that same OS, or on the other Windows OSs because they both use the Windows API. *See* Ex-1003, ¶165. Schaefer explicitly references Win32 API. Ex-1008, [0005], [0012], [0052]. It would also be obvious to incorporate Gaines’ translation layer into Schaefer, allowing an application to be run on different operating systems such as Windows and Unix, for the same reasons discussed in Section VII.A.1.c regarding McMillan. *See* Ex-1003, ¶166.

d. 1[a]

Limitation 1[a] is obvious. Ex-1003, ¶167.

First, Schaefer’s multiple computer deployment renders obvious “a plurality of secure containers of application software,” where “each container compris[es] one or more of the executable applications and a set of associated system files required to execute the one or more applications.” As previously discussed, Schaefer teaches that its virtual environments provide a private contexts for each application to run, which a POSITA would understand includes executable applications (*e.g.*, Microsoft Word or Excel). *See* Ex-1008, [0013], [0019]. Schaefer also explicitly teaches that

the OS Guard provides a virtual environment manager, which includes virtual files, configuration, and shared objects, and a virtual file system manager:

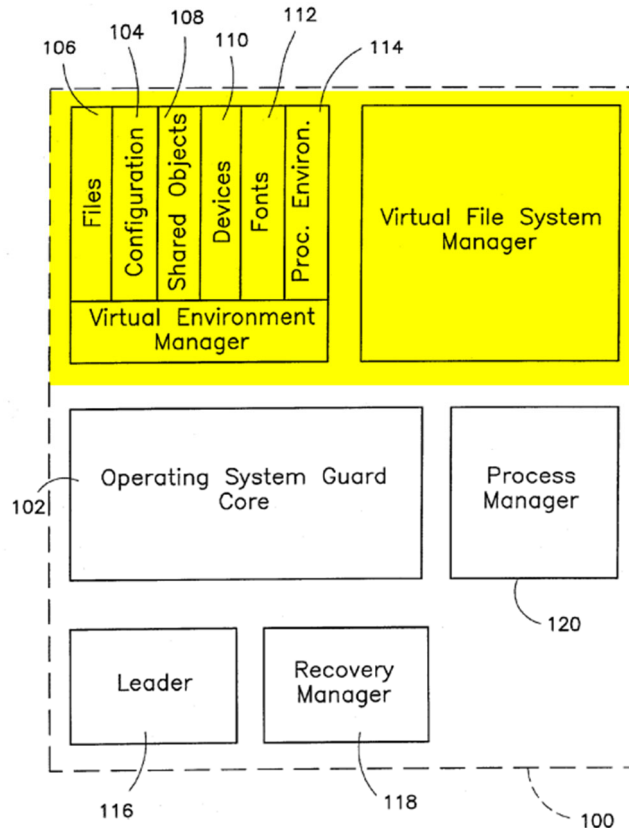


Fig. 4

Ex-1008, Fig. 4 (annotated); Ex-1003, ¶170. These files within the virtual environment manager, including virtual files and configuration, and virtual file system manager, in addition to the application, are containers, which are “[a]n aggregate of files required to successfully execute a set of software applications on a computing platform.” Ex-1001, 2:23-25. Schaefer also explains that the OS Guard enables each application to run within a private context of the system (annotated), with differing copies of system files, configuration, and data. Ex-1008, [0019]; *see*

also id., [0022–0063], [0040], Fig. 2, Fig. 4 (virtual environment manager with “files” and “configuration”); Ex-1003, ¶¶168–170.

Indeed, PO conceded to the EPO that Schaefer’s OS Guard “provided, for each application, more or less *duplicates* of the system files of the operating system” and “each application sees its own modified system files.” Ex-1013 (EP FH), 150. Thus, each private context is a secure container, comprising one or more of the executable applications and a set of associated system files required to execute the one or more applications. Ex-1003, ¶¶171-172. A POSITA would have further understood and found obvious that the virtual environments (private contexts) are “secure containers of application software” under the parties’ agreed construction “[a] container environment where each application set appears to have individual control of some critical system resources and/or where data within each application set is insulated from effects of other application sets,” wherein “each container for use on a server is mutually exclusive of the other containers, such that read/write files within a container cannot be shared with other containers.” Specifically, Schaefer teaches that the virtual environments provide private contexts for applications to run, which localize all changes to the virtual environment, and appears to have its own control of resources/data, as opposed to actual changes to the underlying OS or other containers. *See* Ex-1008, [0013]; *see also id.*, [0063] (OS Guard “keeps the two instances of the application 50 from interfering with one

another”). These virtual environments/private contexts are mutually exclusive of other containers, such that read/write is not shared as Schaefer discloses applications are isolated by the OS Guard, such that the OS Guard acts as a barrier. *See* Ex-1008, [0012], [0019-0020], [0044], Figs. 1-2; Ex-1003, ¶¶172–174. Thus, a POSITA would understand that Schaefer’s OS Guard provides for virtual environments or private contexts (containers), wherein each application running in a private context is insulated from others, and read/write files within a container are not shared with other containers. Ex-1003, ¶¶172–174.

Second, Schaefer’s multiple-computer embodiment renders obvious “storing in memory accessible to at least some of the servers” the plurality of secure containers (*i.e.*, virtual environments/private contexts). As previously discussed, Schaefer discloses that “[t]he present invention is particularly useful for applications delivered over a network” and thus, a POSITA would have found it obvious that because multiple Schaefer computers can be networked, any computer can access another container, which would be stored in memory on the network, for example on one or more computer(s) on the network. Ex-1008, [0042]; Ex-1003, ¶175.

Additionally, a POSITA would have found it obvious that any of Schaefer’s computers could transfer and/or download virtual environments (private contexts) to any other. Ex-1003, ¶176. For example, Schaefer’s OS Guard includes a component

“loader subsystem 116” (red)⁴ which “is used to allow virtual environments to be transferred into and out of the running system.” Ex-1008, [0024].

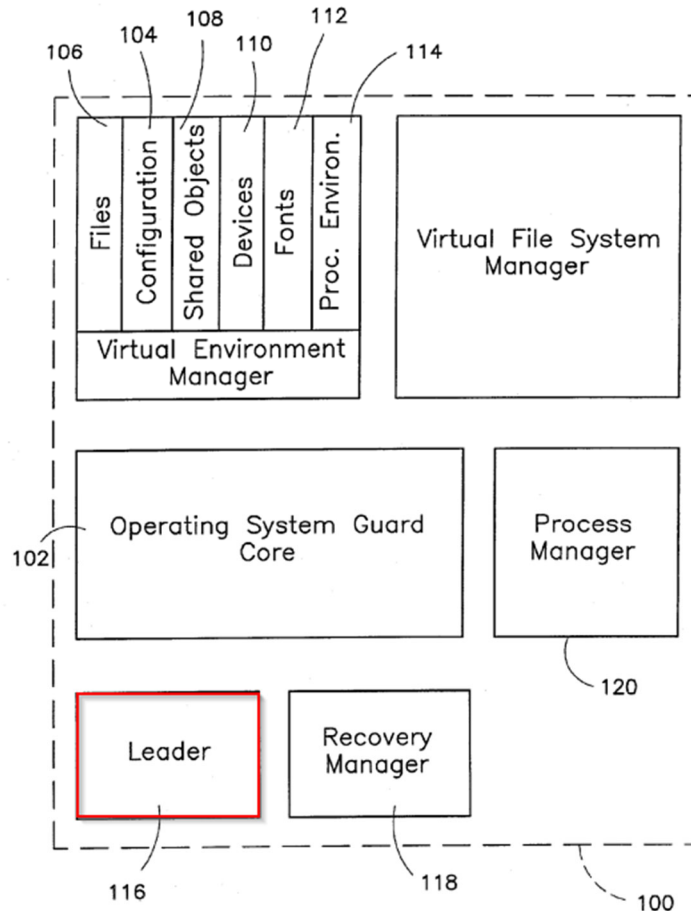


Fig. 4

Ex-1008, Fig. 4 (annotated). A POSITA would thus have understood and found it obvious that the virtual environments/private contexts created by the OS Guard (containers) would thus be stored in memory, and could be loaded and unloaded into

⁴ As Dr. Mowry explains, based on Schaefer’s specification, “leader” in Figure 4 is a typographical error and should read “loader,” and was corrected in the issued patent Ex-1020 (7,028,305). Ex-1003, ¶176.

a given networked computer via Schaefer's loader subsystem. Ex-1003, ¶¶176–177. Gaines' distributed execution environment similarly provides the ability to transfer programs amongst computers over the network, and it would be obvious to apply that teaching to Schaefer's networked environment in the same manner because it would amount to a simple substitution of known prior art elements and is consistent with the express teachings of Schaefer. *See* Ex-1007, 13:40-44, Fig. 3A; Ex-1003, ¶178. It would have been obvious to incorporate this teaching into Schaefer given the similar architectures, and given that the ability to transfer programs between computers in a networked environment improves operation of the overall system by making programs available to multiple computers in the network. This would at a minimum be use of known technique to improve similar systems in the same way. Ex-1003, ¶178.

Third, Schaefer's multiple-computer networked environment renders obvious that the secure containers are “for use with a local kernel residing permanently on one of the servers.” Schaefer teaches that the virtual environment/private contexts (*i.e.*, secure containers of application software) provided by the OS Guard are primarily (though not exclusively) for use with a given computer with a base OS. Specifically, Schaefer explains (and illustrates in Fig. 1) that certain commands pass through the OS Guard “and are then transferred to the Operating System itself.” Ex-1008, [0063]. A POSITA would understand that the underlying OS (10) includes

the kernel, and thus the virtual environment/private contexts provided by the OS Guard are for use with the local kernel residing permanently on any given computer on Schaefer's networked environment. Ex-1008, Fig. 1; Ex-1003, ¶¶179–180.

e. 1[b]

Limitation 1[b] is obvious. Ex-1003, ¶181. For example, consider again a configuration of five (5) networked Schaefer computers, with two UNIX computers, two Windows NT computers, and another Windows 2000 computer. In a private context (container) designed for operation with a base Windows NT OS, the system files would be compatible with the local kernel of the computers running Windows NT as a base OS. The OSs on these Windows NT servers are “at least some of the plurality of different operating systems.” Ex-1003, ¶182. Additionally, the set of associated system files would also be compatible with the local kernel of the computers running Windows 2000 as a base OS, as both Windows NT and Windows 2000 would utilize Win32 API. *See* Ex-1003, ¶182; Ex-1009, Fig. 11-4 (“The Win32 API allows programs to run on almost all versions of Windows.”), (illustrating Windows NT and Windows 2000).

Further, as discussed previously regarding McMillan, Gaines expressly discloses a “virtual operating system is capable of executing programs in a standardized programming language...to provide the remote user *with the ability to run programs that are host-independent.*” Ex-1007, 4:13-18. A POSITA would

be motivated to use Gaines’ “language engine” (and “virtual application interface” and “resource filters”) with Schaefer to achieve these same benefits for all of the same reasons discussed in Section VII.A.1.e regarding McMillan. *See* Ex-1003, ¶¶183–184. That section is incorporated by reference as if fully set forth herein.

f. 1[c]

Limitation 1[c] is obvious. Ex-1003, ¶185. First, Schaefer’s private contexts run on top of an operating system where the kernel is located, and therefore exclude a kernel. Ex-1003, ¶186. As previously discussed, Schaefer’s private contexts (including copies/modified copies of associated system files) need to be compatible with the various OSs (including local kernel) on which the containers are running, including because Schaefer explicitly discusses that certain commands pass through the OS Guard “and are then transferred to the Operating System itself.” Ex-1008, [0063]. A POSITA would understand that because the virtual environment/private contexts communicate with the underlying base OS, the virtual environment/private contexts do not include a kernel. Ex-1003, ¶186. Indeed, consistent with the EPO’s findings, Schaefer’s Figure 1 discloses this element, which PO did not contest to the EPO. *See* Ex-1013 (EP FH), 138.

Moreover, Schaefer discloses (and PO conceded to the EPO) that each container (virtual environment/private context) has its own copies/modified copies

of base OS system files utilized in place of the underlying base OS files. Ex-1008, [0019], [0027], Fig. 4; Ex-1013 (EP FH), 150; Ex-1003, ¶¶187–188.

g. 1[d]

Limitation 1[d] is obvious. Ex-1003, ¶189. As previously discussed, Schaefer’s virtual environments/private contexts (containers) utilize copies/modified copies of the underlying base OS file system, which the PO conceded to the EPO. Ex-1008, [0019], [0027], Fig. 4; Ex-1013 (EP FH), 150; Ex-1003, ¶¶190–191.

h. 1[e]

Limitation 1[e] is obvious. Ex-1003, ¶192. As previously discussed, Schaefer teaches that the private contexts (containers) provided by the OS Guard provide secure environments such that the applications are constrained to that particular context. *See* Ex-1008, [0013], [0015], [0019]. Thus, in Schaefer, the software for a particular application software cannot be shared between the plurality of secure containers. Ex-1003, ¶193.

Additionally, a POSITA would have understood and found it obvious that each of the containers has its own unique root file system that is different from the OS’s root file system. Ex-1003, ¶194. Schaefer explicitly teaches that the virtual environment/private contexts provide a virtual file system, which is unique and different from the underlying OS’s file system. Ex-1008, [0040]; *see supra* Claim

1[d]; Ex-1003, ¶194. Indeed, this is consistent with the EPO’s findings that Schaefer explicitly discloses a unique root file system based on the virtual file system, which the PO did not contest. *See* Ex-1013 (EP FH), 213.

2. Claim 2

Claim 1 is obvious. *See supra* Claim 1. Claim 2 is also obvious. Ex-1003, ¶195.

As previously discussed, Schaefer teaches that different applications can **run** in private contexts of the system. *See* Ex-1008, [0019]; *see also id.*, Cl. 7 (OS Guard “manages the integration of multiple instances of an application by recognizing how many instances of an application are running.”); Ex-1003, ¶196. A POSITA would thus understand that for an application to run, an execution file would be required to start the application within the private context. Ex-1003, ¶197. Compiled object code, such as applications, are made to run through the use of an executable file. *See* Ex-1009, 498 (“For (execute only) program text, the **executable binary file (i.e., .exe or .dll file)** ...”); *see also id.* at 267; Ex-1010, 5 (“Each executable image (.EXE)...”); Ex-1003, ¶197.

3. Claim 6

Claim 2 is obvious. *See supra* Claim 2. Claim 6 is also obvious. Ex-1003, ¶198.

First, Schaefer renders obvious that each container has a unique identity. As previously discussed, Schaefer teaches that its “loader subsystem 116” “is used to allow virtual environments to be transferred into and out of the running system.” Ex-1008, [0024]. A POSITA would have found it obvious that in order to load/unload virtual environments, the containers of application software would need a form of unique identity. Ex-1003, ¶199. Indeed, PO acknowledged to the EPO that Schaefer at least taught that “each application may be attributed some task number,” *i.e.*, as a way to identify the applications. *See* Ex-1013 (EP FH), 148. Consistent with Schaefer’s disclosure, Gaines also teaches that a unique identity is assigned to particular copies of the virtual OS. *See* Ex-1007, 11:17-22. Thus, a POSITA would have found it obvious that a unique identity would be assigned to each of the private contexts in Schaefer’s system, such that the OS Guard can identify the private contexts to load. Ex-1003, ¶200.

Second, a POSITA would have found it obvious to include in the unique identity an IP address, hostname, and/or MAC address. All of these were common prior art methods to identify resources on a computer and a computer network, and are not inventions of the ’814 Patent. Ex-1003, ¶201. Schaefer discloses a networked implementation. *See* Ex-1003, ¶201; Ex-1009, 420. In a network configuration, a POSITA would have found it obvious that the unique identity needs to include some way(s) for a computer seeking access to the container to locate and communicate

with the central server on which is it stored. In any such network, the machines on the network would have names and addresses. Thus, any container on the central server would have an associated name and network address. One common network at the time was an IP network, which uses IP addresses to identify devices and other resources on the network. It would have been obvious to a POSITA to use this well-known network in Schaefer so that uniquely identifiable containers could be located on the network and downloaded to a computer seeking to use the container. *See* Ex-1003, ¶202; Ex-1009, 420. Indeed, Gaines expressly references use of protocols (Telnet and FTP) that use IP addresses. Ex-1007, 6:8-21; Ex-1003, ¶202.

This would also be obvious in view of Tormasov, which as discussed in Ground I, discloses virtual environments in an IP network that have an associated IP address. *See* Ex-1011, Fig. 1, [0024]. As Dr. Mowry explains, and as discussed in Ground I, a POSITA would have found it obvious to apply Tormasov's teachings of a unique identity including an IP address to each of Schaefer's containers for the same reasons as applying them to the virtual environments of McMillan. *See supra* VII.A.3; Ex-1003, ¶¶203–204.

Additionally, a POSITA would have understood that MAC addresses were well-known to identify particular devices when connected within a local network. Ex-1003, ¶205; Ex-1019, 1:6-2:11. A POSITA would have also understood that hostnames were common through use of remote computers and included in

Windows machines as discussed above. Ex-1003, ¶205. As previously discussed for Ground I, a POSITA would have found it obvious and beneficial that each private context's (container's) IP address, MAC address, and/or host name would form part of the unique identity, such that each computer can identify and load a given virtual environment/private context as the plurality of servers are connected over the Internet (or local network), and thus utilize each virtual OS environment's (container's) IP address, MAC address, and/or hostname as part of the unique identity. Ex-1003, ¶206; *see supra* VII.A.3. The EPO similarly found that the use of a unique identity with these network parameters could not have been inventive, as using “the incorporation of network parameters ... into an identifier is a common practise [sic] used to identify resources (including services) across global networks.” Ex-1013 (EP FH), May 5, 2006 Rejection, 140.

4. Claim 9

Claim 2 is obvious. *See supra* Claim 2. Claim 9 is also obvious. Ex-1003, ¶207.

For example, Schaefer teaches that the system “continually monitors the use of shared system resources and acts as a service to apply and remove changes to system components.” Ex-1008, [0004]; *id.*, Cl. 4. A POSITA would understand that “shared system resources” would include hardware resources such as CPU memory, network bandwidth, and disk allocation, and that this information is to be used by

executing application. Ex-1003, ¶208. Indeed, the EPO similarly found that this disclosure discloses this claim element. *See* Ex-1013 (EP FH), 141. Additionally, as discussed above, each of Schaefer's private contexts (containers) include a virtual file system. As Dr. Mowry explains, this information provides disk allocation information. Ex-1003, ¶209. Gaines similarly teaches constraints on resources to be made available to given applications. *See* Ex-1007, 13:52-57; Ex-1003, ¶210. Thus, given Schaefer's express teachings, a POSITA would have understood and found it obvious to associate a portion of CPU memory, network bandwidth, and/or disk allocation when loading the Schaefer's containers on a given computer system prior to application execution, otherwise the application could not run on a given system. Ex-1003, ¶210.

5. Claim 10

Claim 2 is obvious. *See supra* Claim 2. Claim 10 is also obvious. Ex-1003, ¶211.

Schaefer explains that its virtual environments prevent changes to the underlying OS, and any changes are only made in the virtual environment during execution of the application. *See* Ex-1008, [0013], [0015], [0019], [0038], [0041], [0052], Cl. 2. Thus, a POSITA would understand that the applications running within a private context of a virtual environment have no access to system files or applications in other containers or the underlying base OS system files during

execution, since they are part of the client computer. Ex-1003, ¶¶212–214; *see* Ex-1008, [0012], [0017], [0063], Figs. 1-2. This is consistent with the EPO’s findings that Schaefer’s private contexts provide a container with no access to system files of applications in other containers. *See* Ex-1013 (EP FH), 141.

VIII. SECONDARY CONSIDERATIONS

In a boilerplate discovery response served in the Related IBM Litigation, PO alleged that commercial success and long-felt need are secondary considerations that support non-obviousness of the Challenged Claims. Ex-1016, 11-15. However, PO offers only conclusory arguments of commercial success and long-felt need with no evidence to support its allegations, including no showing of a nexus between the alleged commercial success/long-felt need and the claimed invention. *Id.* Petitioner also disputes in the Related IBM Litigation that it has ever practiced the claimed invention. As Dr. Mowry explains, the asserted prior art demonstrates that there was no long felt need for the claimed inventions because the patented features were already well known in the prior art, nor is Dr. Mowry aware of any evidence of commercial success. Ex-1003, ¶216. Moreover, the alleged secondary considerations cannot overcome the strong *prima facie* case of obviousness shown in this Petition. *See Wyers v. Master Lock Co.*, 616 F.3d 1231, 1246 (Fed. Cir. 2010).

Should PO attempt to meet its burden to show secondary considerations, Petitioner reserves the right to respond.

IX. PTAB DISCRETION SHOULD NOT PRECLUDE INSTITUTION

The Board should institute review of the Challenged Claims.

A. Discretionary Denial Under 35 U.S.C. § 325(d) is Not Warranted

All six Becton Dickinson factors weigh in favor of institution. *Becton, Dickinson, & Co. v. B. Braun Melsungen AG*, IPR2017-01586, Paper 8 (PTAB Dec. 15, 2017); *see also Advanced Bionics, LLC v. Med-El Elektromedizinische Geräte GmbH*, IPR2019-01469, Paper 6 at 8 (PTAB Feb. 13, 2020). The Board has consistently “held that a reference that ‘was neither applied against the claims nor discussed by the Examiner’ does not weigh in favor of exercising [] discretion under § 325(d).” *Fasteners for Retail, Inc. v. RTC Indus., Inc.*, IPR2019-00994, Paper 9 at 7-11 (PTAB Nov. 5, 2019). Here, while Schaefer was disclosed in an IDS, Schaefer was not substantively discussed or relied upon by the Examiner. Moreover, none of the grounds in this Petition were evaluated during prosecution. The Examiner incorrectly concluded that the Challenged Claims were not anticipated or obvious because the Examiner did not consider the references and combinations presented herein. *Bowtech Inc. v. MCP IP, LLC*, IPR2019-00383, Paper 14 at 5 (PTAB Aug. 6, 2019) (petitioner did not need to explain how the Examiner erred “because the Examiner did not consider the combinations of the references asserted in the Petition at all”).

B. Discretionary Denial Under 35 U.S.C. § 314(a) is Not Warranted

Evaluation of the factors under *Apple Inc. v. Fintiv, Inc.*, IPR2020-00019, Paper 11 (P.T.A.B. Mar. 20, 2020) (precedential) and the clarifications made by the Director of the USPTO favors institution. See USPTO Memorandum, Interim Procedure for Discretionary Denials in AIA Post Grant Proceedings with Parallel District Court Litigation, 2 (June 21, 2022) (“Director’s Memo”).

To avoid overlap with parallel district court proceedings, and consistent with *Sotrea Wireless, Inc.*, Petitioner hereby stipulates “that, if the PTAB institutes *inter partes* review, Petitioner ‘will not pursue in [*VirtaMove, Corp. v. International Business Machines Corp.*, No. 2:24-cv-00064-JRG (E.D. Tex.) (the “Related IBM Litigation”)] the specific grounds [asserted in the *inter partes* review], or on any other ground ... that was raised or could have been reasonably raised in an IPR (i.e., any ground that could be raised under §§ 102 or 103 on the basis of prior art patent or printed publications).” *Sotera Wireless, Inc. v. Masimo Corp.*, IPR2020-01019, Paper 12 at 13-14 (PTAB Dec. 1, 2020) (precedential as to § II.A).

Other relevant factors weigh against discretionary denial under § 314(a). First, it is early in the district court case. No claim construction opinions have been issued, no depositions on the merits have been taken, and no expert reports have been submitted. Second, while the Board’s final written decision may precede the trial date in the district court case, as jury selection is scheduled for October 20,

2025, the likelihood of postponement is increased given that there are 13 cases scheduled for jury selection on that date.

Additionally, as demonstrated in Section VII, the Petition presents “compelling” arguments for unpatentability of the Challenged Claims based on the same features and functionality described in the ’814 Patent specification. Director’s memo at 4-5 (“the PTAB will not deny institution based on *Fintiv* if there is compelling evidence of unpatentability”); *CommScope Techs. LLC v. Dali Wireless, Inc.*, IPR2022-01242, Paper 23 at 3-5 (Feb. 27, 2023) (precedential) (evidence is compelling when it is “highly likely that the petitioner would prevail with respect to at least one challenged claim”). Thus, institution is consistent with the significant public interest against “leaving bad patents enforceable.” *Thryv, Inc v. Click-To-Call Techs., LP*, 140 S. Ct. 1367, 1374 (2020).

X. MANDATORY NOTICES

A. Real Party-In-Interest

International Business Machines Corp. and Red Hat, Inc. are the real parties-in-interest.

B. Related Matters

The ’814 Patent is the subject of multiple parallel proceedings in the Northern District of California, Eastern District of Texas, and Western District of Texas. *VirtaMove, Corp. v. Hewlett Packard Enterprise Co.*, No. 2:24-cv-00093 (Lead Case); 2:24-cv-00064 (Member Case) (E.D. Tex.); *VirtaMove, Corp. v.*

Amazon.com, Inc., No. 7:24-cv-00030 (W.D. Tex); *VirtaMove, Corp. v. Google LLC*, No. 7:24-cv-00033 (W.D. Tex.); *VirtaMove, Corp. v. Microsoft Corp.*, No. 7-24-cv-00338 (W.D. Tex.); *VirtaMove, Corp. v. Oracle Corp.*, 7-24-cv-00339 (W.D. Tex.); *Red Hat, Inc. v. VirtaMove, Corp.*, No. 3:24-cv-04740 (N.D. Cal.). *VirtaMove, Corp v. Amazon.com, Inc.* and *VirtaMove, Corp. v. Google, LLC* have been ordered to be transferred to the Northern District of California. The '814 Patent is also subject to several proceedings before the Patent Office: *Amazon.com, Inc. v. VirtaMove, Corp.*, IPR2025-00566, IPR2025-00563 and *Google, LLC v. VirtaMove, Corp.*, IPR2025-00488, 00487. These actions may affect, or be affected by, decisions in this proceeding. Nothing herein should be taken as an endorsement of VirtaMove's positions in the co-pending litigations.

C. Counsel Information

Lead Counsel	Back-Up Counsel
James E. Marina, P.C. Reg. No. 41,969 jmarina@kirkland.com <u>Postal and Hand-Delivery Address:</u> KIRKLAND & ELLIS LLP 601 Lexington Ave., New York, NY 10022 Telephone: (212) 446-4877	Todd M. Friedman, P.C. Reg. No. 42,559 tfriedman@kirkland.com <u>Postal and Hand-Delivery Address:</u> KIRKLAND & ELLIS LLP 601 Lexington Ave., New York, NY 10022 Telephone: (212) 446-4786 Brandon H. Brown, P.C. <i>pro hac vice</i> motion filed concurrently bhbrown@kirkland.com <u>Postal and Hand-Delivery Address:</u> KIRKLAND & ELLIS LLP

Lead Counsel	Back-Up Counsel
	555 California St., San Francisco, CA 94104 Telephone: (415) 439-1670

D. Service Information

IBM concurrently submits a Power of Attorney, 37 C.F.R. § 42.10(b), and consents to electronic service directed to the following email address: IBM-VirtaMoveIPR@kirkland.com.

XI. PAYMENT OF FEES

The undersigned authorizes the Office to charge the fee set forth in 37 C.F.R. § 42.15(a)(1) any additional fees that may be due in connection with this Petition to Deposit Account No. 506092.

XII. CERTIFICATION

Petitioner certifies, pursuant to Rule 42.104(a), that the '814 Patent is available for IPR and that Petitioner is not barred or estopped from requesting an IPR of the Challenged Claims on the grounds identified in this Petition. Petitioner certifies: (1) Petitioner is not the owner of the '814 Patent; (2) Petitioner (or any real party-in-interest) has not filed a civil action challenging the validity of any claim of the '814 Patent; (3) this Petition is being filed within one year of service of a complaint against Petitioner asserting the '814 Patent; (4) estoppel provisions of 35

U.S.C. § 315(e)(1) do not prohibit this IPR; and (5) this Petition is filed after the '814 Patent was granted.

XIII. CONCLUSION

Petitioner requests institution of an IPR and cancellation of the Challenged Claims.

Dated: February 7, 2025

Respectfully submitted,

/s/ James E. Marina
James E. Marina, P.C.
Reg. No. 41,969

CERTIFICATE OF COMPLIANCE

This Petition complies with the type-volume limitations as mandated in 37 C.F.R. § 42.24, totaling 13,626 words. Counsel has relied upon the word count feature provided by Microsoft Word.

/s/ James E. Marina
James E. Marina, P.C.
Reg. No. 41,969

CERTIFICATE OF SERVICE UNDER 37 C.F.R § 42.6 (E)(4)

The undersigned hereby certifies that a copy of the foregoing document with accompanying exhibits, as listed in Petitioner's Exhibit List in this document, was served via overnight Federal Express directed to the attorney/agent of record for the patent at the following address:

Allen, Dyer, Doppelt + Gilchrist, PA
1135 East State Road 434, Suite 3001
Winter Springs, FL 32708

A courtesy copy was also served via electronic mail on the attorneys of record listed below for the following related matter.

Reza Mirzaie (CA SBN 246953)
rmirzaie@raklaw.com
Marc A. Fenster (CA SBN 181067)
mfenster@raklaw.com
Neil A. Rubin (CA SBN 250761)
nrubin@raklaw.com
Amy E. Hayden (CA SBN 287026)
ahayden@raklaw.com
Christian W. Conkle (CA SBN 306374)
cconkle@raklaw.com
Jonathan Ma (CA SBN 312773)
jma@raklaw.com
Daniel Kolko (CA SBN 341680)
dkolko@raklaw.com
rak_virtamove@raklaw.com
RUSS AUGUST & KABAT
12424 Wilshire Boulevard, 12th Floor
Los Angeles, CA 90025
Telephone: (310) 826-7474

Qi (Peter) Tong (TX SBN 24119042)
ptong@raklaw.com
RUSS AUGUST & KABAT
4925 Greenville Ave, Suite 200
Dallas, TX 75206

Petition for *Inter Partes* Review of U.S. Patent No. 7,519,814
Case IPR2025-00599

/s/ James E. Marina
James E. Marina, P.C.
Reg. No. 41,969