

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

AMERICAN AIRLINES, INC.,

and

SOUTHWEST AIRLINES, CO.,

Petitioners,

v.

INTELLECTUAL VENTURES II LLC,

Patent Owner

Case No. IPR2026-00043

U.S. Patent No. 7,721,282

PATENT OWNER'S PRELIMINARY RESPONSE

TABLE OF CONTENTS

I.	INTRODUCTION	1
II.	THE SPECIFICATION OF THE ‘282 PATENT	4
III.	THE FILE HISTORY OF THE ‘282 PATENT	7
IV.	THE CLAIMS OF THE ‘282 PATENT	10
V.	CLAIM CONSTRUCTION.....	11
VI.	REFERENCES RELIED ON IN THE PETITION	15
A.	Birse (EX1005).....	15
B.	Declaration of Erez Zadok (EX1003).....	18
1.	Example 1: The expert report collapses claim-defined "leaf image" functionality into Birse's "modified by the user" shadow volumes without addressing the node-specific requirement.....	20
2.	Example 2: The expert report "creates" disclosure by importing claim language into Birse and labeling it as prior-art fact	21
3.	Example 3: The expert report converts Birse’s user-preference overlay into “block merging” without technical explanation	23
4.	Example 4: The expert report supplies the “persistent mapping” limitation through generalized inevitability reasoning, not Birse’s disclosure	24
5.	The expert declaration should be afforded little weight under <i>Xerox</i>	26
VII.	PETITIONER HAS NOT MET ITS BURDEN TO ESTABLISH OBVIOUSNESS UNDER 35 U.S.C §103	26
A.	The Petition Fails To Explain How Birse Discloses A Union Block Device That Comprises A Low-Level Driver For Interfacing Between The First And Second Storage Units And The File System Of The Compute Node.....	26
1.	The Petition’s Birse mapping proves redirection to server-side storage, not interfacing between two storage units and the compute node’s file system	27

2.	Petitioner never identifies, in Birse, the claim-required “first storage unit” and “second storage unit” that the low-level driver interfaces with on the compute node side of the file system	29
3.	The Petition is defective because it fails to map Birse to Element [1.6].....	30
B.	The Petition Fails To Explain How Birse’s User Specific “Shadow” Copies Are the Claimed Node-Specific “Leaf Images”	31
1.	According To Petitioner Southwest, the ’282 Patent’s “leaf image” is compute-node- specific and captures node specific block deltas.....	31
2.	Birse’s “shadow” volumes are, by contrast, user specific and intended to roam across clients	32
3.	The Petition’s mapping improperly collapses user roaming overlays into node-specific leaf images	33
4.	Petitioners' Expert Confirms the Deficiency in the Petition's Attempted Mapping Of Birse To The Claimed "Leaf Images"	34
C.	Petitioner’s Inconsistent Identification of the Claimed “Application Environment” Undermines Its Ground 1 Mapping.....	35
1.	The intrinsic record ties “application environment” to a specific claim-required output created by block-level root/leaf merging	37
2.	The Petition maps “application environment” one way for the “distributing” limitation, then a different way for the “creates ... by merging” limitation	38
3.	The expert declaration compounds the inconsistency by equating the “application environment” with stored operating system volumes while defining “merging” as preference merging with a “system environment”	39
4.	These inconsistencies are dispositive because Petitioner never identifies a single, claim-matching “application environment” in Birse that is both distributed and created by the recited merging operation	40

D.	Petitioner fails to explain why Birse’s “merging user preferences” is the claimed “merging blocks” of a root image and a leaf image	41
1.	“User preferences” are not “blocks of a leaf image,” and Petitioner does not bridge that gap.....	42
2.	Birse’s “dynamic merge” can be accomplished at a semantic or user-profile level, not a block-image level	43
3.	The Petition’s own discussion shows redirection or selection, not “merging blocks ... to create the application environment”	43
4.	The Board should reject Petitioner’s conclusory equivalence between “preference merge” and “block merge”	44
E.	The Additional Grounds All Depend On The Deficient Mapping Of Birse To The Independent Claims In Ground 1	45
1.	Ground 2 (Birse + Ballew): Ballew Adds Only HPC or Grid Context and Does Not Address the Deficiencies In Ground 1 With Respect To The Independent Claims	45
2.	Ground 3 (Birse + Wu): Wu Is Offered Only for a “Block Modification Log,” and the Petition Concedes the Key Architecture Still Depends on Birse	46
3.	Ground 4 (Birse + Kleinschnitz): Kleinschnitz Is Offered Only for Snapshot Collapsing and Partitioning, Not for the Missing Birse Elements	47
4.	Grounds 2–4 Do Not Independently Establish Unpatentability	48
VIII.	CONCLUSION	49

PATENT OWNER'S EXHIBIT LIST

EX2001	<i>American Airlines, Inc. and v. Intellectual Ventures II LLC</i> , IPR2025-00785 and IPR2025-00786, Paper 11 (PTAB August 29, 2025) (declining to exercise discretion to deny IPRs).
EX2002	<i>American Airlines, Inc. and v. Intellectual Ventures II LLC</i> , IPR2025-00782 and IPR2025-00931, Paper 12 (PTAB September 19, 2025) (declining to exercise discretion to deny IPRs).
EX2003	<i>American Airlines, Inc. and v. Intellectual Ventures II LLC</i> , IPR2025-00987 and IPR2025-01055, Paper 9 (declining to exercise discretion to deny IPRs).
EX2004	Defendant Southwest Airline Co.'s Preliminary Invalidity Contentions
EX2005	“What’s inside IV’s patent portfolio?” by Kent Richardson and Erik Oliver
EX2006	<i>American Airlines, Inc. and v. Intellectual Ventures II LLC</i> , IPR2025-01055, Paper 11 (PTAB November 21, 2025) (denying institution of IPR)
EX2007	Abstract of Title For U.S. Patent No. 7,721,282
EX2008	Defendant Southwest Airline Co.'s Opening Claim Construction Brief For U.S. Patent Nos. 7,721,282, 8,332,844, 7,949,785, And 8,407,722 in Case No. 7:24-cv-00277, Doc. #76 (WD Tx).
EX2009	Defendant American Airlines, Inc.'s Supplemental Rule 4-2 Disclosures in Case No. 4:24-cv-00980.

I. INTRODUCTION

Petitioner seeks inter partes review of claims 1 through 30 of U.S. Patent No. 7,721,282 (the '282 Patent), asserting four obviousness grounds under 35 U.S.C. § 103 based primarily on Birse (EX1005), with Ballew (EX1006), Wu (EX1007), and Kleinschnitz (EX1008) cited only in combination for select dependent limitations. Petition, 1–4. The Petition should be denied because Petitioner has not carried its institution-stage burden to show a reasonable likelihood of prevailing on any challenged claim.

At the core of every ground is the same premise: that Birse allegedly discloses the independent-claim architecture of the '282 patent. However, the Petition fails to demonstrate that several limitations of the independent claims are met by Birse. For example, the Petition fails to demonstrate that Birse discloses a union block device comprising "a low-level driver for interfacing between the first and second storage units and the file system of the compute node," as required by independent claim 1.

In addition, Petitioner Southwest is advancing inconsistent claim constructions here and in the district court with respect to the term "leaf images" in independent claims 1 and 15 of the '282 Patent. Petitioners' expert in these proceedings interpreted the claims according to their "ordinary and customary meaning." EX1003, ¶213. By contrast, in the district court, Petitioner Southwest opposes interpreting "leaf images" according to its plain meaning and asserts that "a

'leaf image' is a node-specific image of the application environment operating at the block level below the file system." Ex. 2008, 12 (emphasis in original). The Petition fails to demonstrate that Birse discloses a node-specific leaf image, as required by Petitioner Southwest's district court claim construction.

The inconsistent claim construction positions of Petitioners do not end with Southwest's construction of "leaf image." While Petitioners' expert in these proceedings was able to interpret the claims according to their "ordinary and customary meaning" (EX1003, ¶21), Petitioner American contends in the district court that the phrases "appropriate persistent mapping" (in independent claims 1 and 15), "the modifying" (in claim 15) and "at an operational level" (in claim 26) are all indefinite. EX2009, 4-5. As Director Squires explained in *Tesla, Inc. v. Intellectual Ventures II LLC*, IPR2025-00340, Paper 18 (November 5, 2025) (informative), absent a sufficient explanation why inconsistent claim construction positions are warranted, a petitioner cannot argue that a claim is indefinite in the district court and at the same time maintain that the same claim should be interpreted according to its plain and ordinary meaning in IPR proceedings. Here, the Petitioners fail to even acknowledge their existence of inconsistent claim construction positions, yet alone explain why inconsistent claim construction positions are warranted. As explained below, this deficiency provides an independent basis for denial of institution due to

the Petition's failure to satisfy its burden to explain how the challenged claims are to be construed under 37 CFR 42.104(b)(3).

The Petition's allegations that Birse discloses the "application environment" recited in the independent claims of the '282 Patent are also defective. With respect to that limitation, the mappings in the Petition are inconsistent and rely on shifting identifications of key claim concepts and conclusory expert assertions instead of record evidence. For example, Petitioner inconsistently characterizes the "application environment" in Birse, alternately treating it as a stored operating-system copy distributed from a server and as a "dynamically merged system environment" based on user preferences, without reconciling those positions or mapping either of the to the claimed cohesive, block-merged image. Petition, 18, 26, 47–53; EX1001, 6:10–16; EX1003, ¶¶217, 255–266. And Petitioner does not establish the claim-required interface topology for the "low-level driver," instead pointing to Birse's generic statement that requests are "redirected at a lower level of the operating system," which is not the same as a driver that interfaces between two image storage units and the compute node's file system as claimed.

Because Grounds 2 through 4 expressly build on the same defective Birse mapping to the independent claims and cite the additional references only for discrete dependent-claim features, those grounds do not cure the Petition's threshold failures. Petition, 74–86. For these reasons, and as set forth below, the Petition does

not meet the statutory standard for institution, and the Board should deny institution of this *inter partes* review.

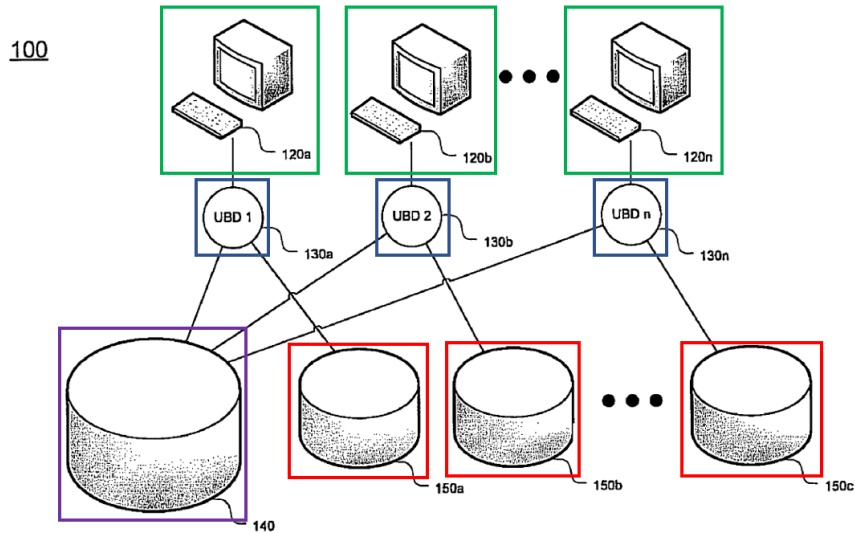
II. THE SPECIFICATION OF THE '282 PATENT

The '282 Patent addresses the problem of deploying and maintaining application environments (including operating-system boot images) across multi-node computing systems such as clustered or high-performance computing environments, where pre-creating and storing a separate full boot image for each compute node can increase bring-up time and waste storage. EX1001, 2:3–12. To reduce deployment friction while still allowing node-specific customization, the patent describes distributing an "application environment" using a shared "root image" and per-node "leaf images," with a "filter" (implemented as a union block device) operating between the compute nodes and the file system to merge node-specific changes into a cohesive environment for a given node. *Id.*, 2:3–5 and 6:10–16.

Architecturally, the system includes a first storage unit that stores blocks of a root image (for example, an operating system) and second storage units that store leaf images comprising new data blocks and changes to blocks of the root image. *Id.*, 4:12–22. Each compute node has a corresponding leaf image that captures the changes made by that node to its instance of the application environment, including

blocks unique to that compute node. *Id.*, 4:17–35. The union block device is implemented as a low-level driver below the file system that "creates the application environment" by merging blocks from the root image with blocks from the leaf image. *Id.*, 7:24–33. The specification emphasizes the practical effect of this approach: the resulting environment provides the compute node with "one cohesive image rather than a base image plus its additions, deletions, and modifications." *Id.*, 6:10–16.

Figure 1 of the '282 Patent (reproduced below with colored annotations added) illustrates a system "for distributing an application environment to one or more compute nodes." *Id.*, 3:62-64. "System 100 has a number of compute nodes 120*a-n* [green boxes] coupled to first storage unit and a corresponding second storage unit 150*a-n* [red boxes] though a corresponding union block device (UBD) 130*a-n* [blue boxes]." *Id.*, 4:3-6. System 100 also "has a first storage unit 140 [purple box] for storing blocks of a root image of an application environment. The root image contains data initially common to the compute nodes 120*a-n*." *Id.*, 4:11-13.



"[E]ach compute node 120a-n has a corresponding second storage unit 150a-n for storing a leaf image. ... [A] leaf image will describe the changes made by the respective compute node 120a-n to its instance of the application environment. Thus, when a compute node (e.g., node 120a) makes changes involving the root image, modifications are made to that compute node's leaf image (e.g., leaf image stored on second storage device 150a)." *Id.*, 4:17-31.

"A compute node 120a-n mounts its instantiation of the application environment via its respective UBD 130a-n. The UBDs 130a-n are effectively low-level drivers that operate as an interface between the first and second storage devices and the file system of each compute node 120a-n." *Id.*, 4:41-45. In one embodiment, the file system "reside[s] on each of the compute nodes 120a-n," and the "low-level driver determin[es] which data blocks are needed for the compute node's instance of the application environment and deliver[s] the application environment to the

compute [node] via the compute node's file system." *Id.*, 4:46-47 and 6:18-22 (emphasis added).

Operationally, the patent focuses on recording node-specific modifications in the leaf image in response to the compute node's use of the environment, enabling the system to avoid storing a full duplicated boot image per node while still permitting node-specific changes and customization. *Id.*, 4:17–35. The patent also describes workflows in which accumulated leaf-image changes can be reconciled and used to update or form new root images, supporting systematic lifecycle management of distributed environments while preserving a cohesive, locally available application environment for a given node. *Id.*, 6:10–16.

III. THE FILE HISTORY OF THE '282 PATENT

The '282 Patent matured from U.S. Patent Appl. 11/395,816 filed March 30, 2006 (the Application). EX1002. As filed, claim 1 of the Application recited as follows:

1. A system for distributing an application environment to a compute node comprising:
 - a first storage unit for storing blocks of a root image of the compute node;
 - a second storage unit for storing a leaf image, the leaf image comprising new data blocks and changes to the blocks of the root image; and
 - a union block device for interfacing between the compute node and the first and second storage units to distribute the application environment to the compute node, wherein the union block device creates the application environment by merging the blocks of the root image stored on the first

storage unit with the blocks of the leaf image stored on the second storage unit. *Id.*, 22.

In the first Office Action, the Examiner rejected all of the claims as obvious in view of U.S. Patent No. 6,502,238 (Pavan) in view of U.S. Patent No. 7,475,274 (Davidson). *Id.*, 65-81. In response, the Applicant amended claim 1 as follows:

1. (Currently amended) A system for distributing an application environment ~~to a compute node~~ comprising:

a compute node comprising a computer system;

a first storage unit for storing blocks of a root image of the compute node, wherein the first storage unit comprises a first non-volatile memory, wherein the root image comprises a computer program, wherein the blocks comprise sections of data, and wherein a file of the root image comprises at least one block;

a second storage unit for storing a leaf image, the leaf image comprising new data blocks and changes to the blocks of the root image, wherein the second storage unit comprises a second non-volatile memory; and

a union block device for interfacing between the compute node and the first and second storage units to distribute the application environment to the compute node, wherein the union block device comprises a driver, wherein the union block device creates the application environment by merging the blocks of the root image stored on the first storage unit with the blocks of the leaf image stored on the second storage unit. *Id.*, 90.

In the accompanying remarks, the Applicant argued, *inter alia*, that "Pavan does not teach or suggest ... a union block device for interfacing between the compute node and the first and second storage units to distribute the application environment to the compute node." *Id.*, 98 (emphasis added).

The Examiner next had a telephone interview with the Applicant during which the Applicant agreed to the addition of the following language new language at the end of claim 1:

the union block device comprises a low-level driver for interfacing between the first and second storage units and the file system of the compute node; and the union block device, upon receiving a write request from the compute node for a sector X, creates an appropriate persistent mapping for sector X. *Id.*, 112 (emphasis added).

Significantly, while the amended claim language previously presented to the Examiner in response to the first Office Action had only recited "a union block device for interfacing between the compute node and the first and second storage units," the new claim language that the Examiner added during the interview required that the union block device comprise a low-level driver "for interfacing between the first and second storage units and the file system of the compute node." *Id.*, 90 and 112. Contemporaneously with the addition of this new claim language, the Examiner allowed the Application. *Id.*, 111-114.

The file history demonstrates that a union block device broadly described as simply "interfacing between the compute node and the first and second storage units" was insufficient from the Examiner's perspective to distinguish claim 1 from the prior art. Rather, as a condition to allowing the Application, the Examiner required that the language of claim 1 go further and specify a union block device for

"interfacing between the first and second storage units and the file system of the compute node."

IV. THE CLAIMS OF THE '282 PATENT

Claim 1 of the '282 Patent is representative, and recites in pertinent part:

1. A system for distributing an application environment comprising:
 - a compute node comprising a computer system;
 - a first storage unit for storing blocks of a root image of the compute node ...;
 - a second storage unit for storing a leaf image, the leaf image comprising new data blocks and changes to the blocks of the root image, ... ; and
 - a union block device ... the union block device comprises a low-level driver for interfacing between the first and second storage units and the file system of the compute node; ... EX1001, 7:12-38 (emphasis added)

Thus, claim 1 recites "a second storage unit for storing a leaf image ... comprising new data blocks and changes to the blocks of the root image." *Id.*, 21-24. As discussed in the next section, Petitioner Southwest contends in the district court that a "leaf image" in the context of claim 1 of the '282 Patent is "a node-specific image of the application environment operating at the block level below the file system." EX2008, 10-12. The Petition fails to explain how the prior art, as relied on the Petition (i.e., Birse) meets Southwest's district court construction of "leaf image."

Claim 1 also recites a union block device for "interfacing between the first and second storage units and the file system of the compute node." EX1001, 7:25-38.

By way of example, this claim limitation corresponds to the disclosure in the specification where the file system "reside[s] on each of the compute nodes 120a-n," and the "low-level driver determin[es] which data blocks are needed for the compute node's instance of the application environment and deliver[s] the application environment to the compute [node] via the compute node's file system." EX1001, 4:46-47 and 6:18-22 (emphasis added). As explained more fully below, the Petition fails to explain how Birse discloses a union block device for "interfacing between the first and second storage units and the file system of the compute node," as claimed.

V. CLAIM CONSTRUCTION

A. "Leaf Image" (Claims 1 and 15)

The Petitioner Southwest is advancing inconsistent claim constructions with respect to the term "leaf images" in independent claims 1 and 15 of the '282 Patent. The Petition contends "that the challenged claims are obvious under any reasonable interpretation under the *Phillips* standard," and that, as a result "the Board need not resolve any express constructions." Petition, 11-12. In keeping with that approach, Petitioner's expert in these proceedings interpreted the claims according to their "ordinary and customary meaning." EX1003, ¶213. By contrast, in the district court, Petitioner Southwest opposes interpreting "leaf images" according to its plain meaning and asserts that "a 'leaf image' is a node-specific image of the application

environment operating at the block level below the file system." EX2008, 12

(emphasis in original). For example, Petitioner Southwest argues as follows with respect to the construction of "leaf image" in the '282 Patent:

... system claim 1 recites "a second storage unit for storing a leaf image, the leaf image comprising new data blocks and changes to the blocks of the root image," and method claim 15 requires "storing a leaf image comprising new data blocks and changes to the blocks of the root image" and "merging the blocks of the root image with the blocks of the leaf image" to "create the application environment" delivered "to the compute node." Thus, the claims specify (i) the operative unit is *blocks* (not files), and (ii) each leaf image is node-specific ("respective ... compute nodes"; "respective leaf images"; "cohesive respective application environments"). ... The Court should therefore adopt Southwest's construction: a "leaf image" is a node-specific image of the application environment operating at the block level below the file system. Id., 11-12 (emphasis in original).

Yet, here the Petition does not argue that a "leaf image" must be "node-specific image of the application environment," nor explain how Birse discloses a "leaf image" that is a "node-specific image of the application environment." As the Director recently ruled in *Revvo Technologies, Inc. v. Cerebrum Sensor Technologies, Inc.*, IPR2025-00632, Paper 20 (Squires November 3, 2025) (precedential), a petitioner seeking to take inconsistent claim construction positions is required to explain why it takes different claim construction positions in the petition and in district court. *See also, Cambridge Mobile Telematics, Inc. v. Sfara, Inc.*, IPR2024-00952, Paper 12 at 8 (PTAB Dec. 13, 2024) (informative). Here, the Petition fails to even acknowledge the existence of inconsistent claim construction

positions, yet alone explain why inconsistent claim construction positions are warranted.

Petitioners' strategy is clear. In the district court, Southwest is advancing a narrow claim construction requiring that a leaf image be "a node-specific image of the application environment" in an attempt to avoid infringement, while relying on a broader claim construction of the term in these proceedings in an attempt to invalidate the patent. That is the epitome of "inconsistent positions" and warrants denial of institution.

In any event, the Petition's decision (at 11) to rest on the assertion that the art meets the claims "under any reasonable interpretation under the *Phillips* standard" has a concrete consequence: Petitioner must prevail under all constructions that Petitioners' deem reasonable including those that Petitioner Southwest is advancing in the district court. Put differently, if there are multiple reasonable interpretations of "leaf image," Petitioners fail to carry their burden if the asserted prior art does not satisfy even one of those reasonable interpretations.

B. "Appropriate Persistent Mappings" (Claims 1 and 15), "The Modifying" (Claim 15) And "At An Operational Level" (Claim 26)

The inconsistent claim construction positions of Petitioners do not end with Southwest's construction of "leaf image." While Petitioner's expert in these proceedings was able to interpret the claims according to their "ordinary and customary meaning" (EX1003, ¶21), Petitioner American contends in the district

court that the phrases "appropriate persistent mapping" (in independent claims 1 and 15), "the modifying" (in claim 15) and "at an operational level" (in claim 26) are all indefinite. EX2009, 4-5. As Director Squires explain in *Tesla, Inc. v. Intellectual Ventures II LLC*, IPR2025-00340, Paper 18 (November 5, 2025) (informative), absent a sufficient explanation why inconsistent claim construction positions are warranted, a petitioner cannot not argue that a claim is indefinite in the district court and at the same time maintain that the same claim should be interpreted according to its plain and ordinary meaning in IPR proceedings. Here, the Petitioners fail to even acknowledge their existence of inconsistent claim construction positions, yet alone explain why inconsistent claim construction positions are warranted.

Petitioners bear the burden of explaining how the challenged claims are to be construed. 37 CFR 42.104(b)(3). The Petitioners here failed to meet that burden because they have advanced inconsistent claim constructions without providing a sufficient explanation why inconsistent claim constructions are warranted. *Revvo Technologies, Inc. v. Cerebrum Sensor Technologies, Inc.*, IPR2025-00632, Paper 20 (November 3, 2025) (precedential) (finding that, where a Petitioner advances different claim constructions across two forums, "the Board erred when it determined that Petitioner complied with Rule 42.104(b)(3) by proposing Patent Owner's district court constructions"); *see also*, the following decisions cited in *Revvo, 10X Genomics, Inc. v. President & Fellows of Harvard College*, IPR2023-

01299, Paper 15 at 13 (PTAB Mar. 7, 2024) and *Orthopediatrics Corp. v. K2M, Inc.*, IPR2018-01546, Paper 10 at 11 (PTAB Feb. 14, 2019) (both denying institution for failing to comply with Rule 42.104(b)(3) where a petitioner, without explanation, proposed different claim constructions before the Board and in district court).

VI. REFERENCES RELIED ON IN THE PETITION

A. Birse (EX1005)

Birse describes a network computer ("NC") system in which one or more NC clients obtain operating system ("OS") functionality from an NC server over a network. Birse explains that an NC server "may provide access to a copy of the operating system to an NC client in response to a boot request," and it depicts NC systems coupled via a network (including via the Internet). EX1005, 4:31–40; EX1005, 14:8–9. Birse's stated objective includes preserving user customizations so that a user's desktop preferences and other customized settings can be supplied when the user logs in, enabling the user to log in to different NC clients while having the same user experience. As Birse explains:

... the NC system may preserve user customizations, such as preferences, by maintaining individual, user, storage areas. When an NC client boots from the network and accesses a stored copy of the operating system from an NC server, the user's preferences are dynamically merged with the system environment provided to the NC client. Advantageously, since, the user's desktop preferences and other customized settings are all preserved from session to session and supplied to the NC client as it boots from the network, the user may login to any NC client on the network and have the same user experience. *Id.*, 2:62–3:3 and 9:2–10 (emphasis added).

Birse further explains that, upon termination of a session, preferences and other user data reflecting changes made during the session may be copied to a user-specific storage location on the server, and then retrieved and supplied to the NC client at the user's next login, whether from the same NC client or a different NC client. As Birse explains:

According to one embodiment, the user environment management process 182 tracks and maintains the persistent user data 186 to insure that changes the user has made during the current session will be persistent at the next login. For example, upon termination of a user session, preferences, desktop items, and various other information representing changes made by the user during the user session are copies to a user-specific storage location on the NC server 170. Subsequently, at the user's next login, from the same or different NC client, the user environment management process 182 will retrieve the data from the corresponding user-specific storage location and return it to the NC client thereby allowing the user to login to any NC client of the NC system 100 and have the same user experience. *Id.*, 6:35–48 (emphasis added).

Birse also describes a copy-on-write "shadow" storage scheme that preserves a baseline OS image while capturing changes separately. In Birse, baseline OS content remains in "system volumes," and modifications made during operation are diverted into corresponding "shadow" storage so the baseline need not be overwritten; Birse then uses mapping constructs (including one-to-one block mapping and band mapping tables) to associate baseline blocks with modified counterparts for purposes of servicing accesses and preserving user customizations. *Id.*, 5:13–14, 7:35–53, 10:63–65, 11:6–17, 11:49–52 and 15:33–37.

In the district court, Petitioner Southwest argues that a "leaf image" in the '282 Patent must be "a node-specific image of the application environment," i.e., the same leaf image is applied for a given node regardless of the user logged into the node. The architecture of Birse is materially different because, in Birse, the so-called leaf images (mapped to, e.g. shadow image 695) are "user-specific images" rather than "node-specific images," which means that a different leaf image is applied to a given node depending on the user logged into the node. The fact that Birse's so-called leaf images are user-specific -- as opposed to node-specific -- is what enables a user in Birse to "login to any NC client on the network and have the same user experience. Petition, 35 (mapping leaf image to, e.g., shadow image 695); EX1005, 2:62–3:3 and 9:2–10 (emphasis added).

Thus, according to the claim construction position advanced by Petitioner Southwest in the district court, the '282 Patent is directed to distributing a compute node's "application environment" by having a union block device create a cohesive image through block-level merging of a shared "root image" with a compute-node-specific "leaf image" stored on a second storage unit corresponding to that compute node. *Id.*, 4:12–22 and 7:24–33. The specification explains that the system "merg[es] the blocks of the root image with the blocks of the leaf image to create the application environment," and that the resulting application environment "will appear to the compute node as one cohesive image rather than a base image plus its additions,

deletions, and modifications." *Id.*, 6:10–16. Thus, whereas Birse frames "shadow" storage as a mechanism for preserving and roaming user-driven changes and preferences alongside baseline system volumes, Petitioner Southwest argues in the district court that the '282 Patent's "leaf image" is expressly tied to the compute node and is part of a distribution mechanism in which the union block device affirmatively "creates" the delivered application environment by merging root-image and leaf-image blocks into what appears to the node as a single cohesive image. EX2008, 11-12; *see also*, EX1001, 4:17–22 and 6:10–16.

B. Declaration of Erez Zadok (EX1003)

Petitioners submit the Declaration of Dr. Erez Zadok (EX1003) as their purported technical evidence to support the Petition's obviousness theories and to supply a POSITA's perspective on how the cited prior art allegedly meets each challenged limitation. However, based on the written record, it appears that Petitioners kept their own expert in the dark with respect to the claim construction positions that Petitioners intended to advance in the district court. Petitioners' expert executed his declaration on Friday, October 10, 2025, based on his understanding that "the terms appearing in the patent claims should be interpreted according to their 'ordinary and customary meaning.'" EX1003, ¶¶213, 415. On the following Tuesday, October 14, 2025, Petitioner Southwest filed its Opening Claim Construction Brief in the district court (EX2008), where Southwest opposed interpreting "leaf images"

according to its plain meaning and asserted that "a 'leaf image' is a node-specific image of the application environment" EX2008, 12 (emphasis in original).

Petitioner Southwest's claim construction position in the district court was not developed overnight, and it would be quite surprising if Petitioner Southwest only initially conceived of its "node-specific" district court claim construction for "leaf image" during the intervening four days between the execution of the Expert Declaration in these proceedings on October 10, 2025 and the filing by Southwest of its Opening Claim Construction Brief in the district court on October 14, 2025. Yet, in the "Materials Considered" section of the Expert Declaration, there is no evidence that Petitioners' Expert considered any materials that would have informed him of the "node-specific" claim construction for "leaf image" that Petitioner Southwest intended to advance in the district court. EX1003, ¶30. Perhaps more importantly, regardless of whether Petitioners' Expert was actually informed that Petitioner Southwest intended to advance a "node-specific" claim construction for "leaf image" in the district court, it is beyond dispute that the Expert Declaration fails to analyze whether Birse discloses a 'leaf image' that is a node-specific image of the application environment," as Petitioner Southwest maintains in the district court.

In a proper petition, an expert declaration should explain the technology, interpret the prior art, and provide reasoned, technically grounded analysis that is

tethered to what the references actually disclose. Here, however, Petitioners use the declaration for an improper purpose: not to illuminate technical facts, but to rehabilitate weaknesses in the Petition by importing claim language into the prior art, relabeling Birse components with bracketed claim terms, and offering conclusory "operates the same way" assertions or generalized "must be present" logic in place of a limitation-specific showing. That is the type of expert "gap filling" and litigation-brief testimony the Board has held should be afforded little weight under *Xerox Corp. v. Bytemark, Inc.*, IPR2022-00624, Paper 9 at 15–17 (PTAB Aug. 24, 2022) (precedential), and it sets the stage for the concrete examples below demonstrating why Zadok's declaration cannot be credited as sufficient to carry Petitioners' institution burden.

1. Example 1: The expert report collapses claim-defined "leaf image" functionality into Birse's "modified by the user" shadow volumes without addressing the node-specific requirement

The '282 patent describes a root image shared across compute nodes and a leaf image "unique to that compute node" that stores "changes made by a compute node to the root image." EX1001, 1:62–67. The Petition and the Expert Declaration attempt to map Birse's user-specific shadow volumes to the "leaf image" by emphasizing that Birse's shadow volume contains portions of the OS "modified by

the user." Petition, 44–45; EX1003, ¶¶252–253, citing EX1005, 5:13–14; EX1005, 11:49–52.

The Expert Report fills the resulting conceptual gap through characterization. It states that Birse "operates the same way," then equates the root image to "system volumes" and the leaf image to "shadow volumes," even though Birse's shadow volumes are described in user-centric terms and used to preserve user-specific changes and preferences. EX1003, ¶¶253, 256–259; Petition, 18, 47–48. The Expert Declaration never explains how Birse's user-modification and user-experience portability framework meets the compute-node-specific leaf image architecture that Southwest asserts is required by the claim language. EX2008, 11-12, *see also*, EX1001, 1:62–67; EX1005, 9:2–10. Instead, the Expert Declaration treats the existence of any CoW shadow mechanism as sufficient, which again is a gap-filling conclusion rather than a limitation-specific showing. EX1003, ¶¶252–257; Petition, 44–45 and 47–48.

2. Example 2: The expert report "creates" disclosure by importing claim language into Birse and labeling it as prior-art fact

Claim 1 recites that "the union block device creates the application environment by merging the blocks of the root image stored on the first storage unit with the blocks of the leaf image stored on the second storage unit." EX1001, 7:24–33; 6:8–16 (Step 330, "merging the blocks ... to create the application environment,"

which appears as "one cohesive image"). The Petition's mapping hinges on the 'assertion in the Expert Report that Birse "operates the same way," and it cites EX1003 as the basis for treating Birse's block device drivers as "union block devices" that "create" and "merge" images. Petition, 47–48; EX1003, ¶¶255–257.

But the Expert Declaration does not actually identify a disclosure in Birse that a driver "creates the application environment" by "merging blocks" of two images in the sense required by the '282 Patent. Instead, the Expert Declaration supplies the missing step by rewriting Birse in claim language and then treating that rewrite as evidence. For example, it states that Birse's block device drivers "(i.e., 'union block devices') 'are configured to create said application environments by merging the blocks of said root image,'... 'with the blocks of respective leaf images.'" EX1003, ¶256. That quoted formulation mirrors the claim requirement, not Birse's actual disclosure, and it is presented as though it were Birse's own teaching. Petition, 47–48 (adopting the same, *ipse dixit*, "operates the same way" conclusion and citing EX1003, ¶¶255–56).

This is impermissible gap filling. The critical question is not whether a declarant can restate the claim using Birse's nouns, but whether Birse actually teaches the claim-required creation of an "application environment" by block-level merging of a root image and a leaf image into a cohesive image. EX1001, 6:8–16; EX1001, 7:24–33. The Expert Declaration does not supply that evidence. It supplies

only an assertion that Birse “operates the same way,” which is the precise kind of ipse dixit *Xerox* holds should be afforded little weight. *Xerox*, IPR2022-00624, Paper 9 at 15–17.

3. Example 3: The expert report converts Birse’s user-preference overlay into “block merging” without technical explanation

The Petition relies heavily on Birse’s “preferences ... dynamically merged with the system environment” language to support “merging” and “application environment.” Petition, 18, 47–48, quoting EX1005, 9:2–10 and 14:61–67. That Birse passage is about preserving a consistent user experience and dynamically applying user preferences, not about creating a cohesive root-plus-leaf block image as required by the '282 Patent. EX1001, 6:8–16 (cohesive image) and 7:24–33 (union block device creates the application environment by merging blocks).

The Expert Declaration fills the resulting analytical gap by treating Birse’s preference behavior as if it were the claimed block-level merge. It quotes Birse’s preference and user-experience statements, including that users may log in to any client and have the same experience, and then simply concludes that this corresponds to the claimed creation of the application environment through root/leaf block merging. EX1003, ¶¶258–259, quoting EX1005, 9:2–10; EX1003, ¶257 (recasting the preference write example as “merging ... to create the application environment

(dynamically merged system environment)"); Petition, 47–48 (adopting the same “dynamically merged system environment” label).

That conclusion is not supported by a reasoned technical explanation of how “preferences dynamically merged” equates to “merging the blocks of the root image” with “the blocks of the leaf image” to create a cohesive image. EX1001, 6:8–16 and 7:24–33. It is instead a relabeling exercise that attempts to bridge an evidentiary gap through mere assertion. Under *Xerox*, such conclusory equivalence testimony is entitled to little weight. *Xerox*, IPR2022-00624, Paper 9 at 15–17.

4. Example 4: The expert report supplies the “persistent mapping” limitation through generalized inevitability reasoning, not Birse’s disclosure

Claim 1 requires that, “upon receiving a write request ... for a sector X,” the union block device “creates an appropriate persistent mapping for sector X.” EX1001, 5:1–11 and 7:24–33. The Petition’s mapping of this limitation relies on the expert report for the key “persistent mapping” step. Petition, 52, citing EX1003, ¶264. Yet the Expert declaration at ¶264 does not cite Birse for “creating an appropriate persistent mapping for sector X” upon receiving a write request. Instead, it advances a generalized inevitability argument, asserting that “any” storage system that maintains OS images and snapshots “must persistently preserve” blocks and that “there must be a mapping of the original vs. modified blocks” or such systems would be “non-functional and useless.” EX1003, ¶264.

That is classic gap filling. The claim does not merely require that data that is stored must be data that is stored, which is a truism. It requires that the union block device, upon receiving a write request for a particular sector X, creates an appropriate persistent mapping for that sector. EX1001, 5:1–11. The reasoning in the Expert declaration does not establish that Birse performs that claim-required act. It merely asserts that some form of mapping must exist in snapshot systems as a matter of general logic, and then treats that generalization as proof that Birse meets the limitation. EX1003, ¶264; Petition, 52.

The Expert Declaration later cites Birse’s redirection of writes into a shadow volume and references mapping concepts such as “one-to-one block mapping” and tables mapping bands, but it again leaps from those disclosures to the specific claim requirement that the union block device “creates an appropriate persistent mapping for sector X,” without establishing that Birse’s mechanism matches either the timing or the function of the claimed mapping creation triggered by the write request for sector X. EX1003, ¶¶266–269, citing EX1005, 7:35–53, 10:63–65, 11:6–17 and 15:33–37); Petition, 52–53 (adopting the same theory and again citing EX1003, ¶¶265–66). Under *Xerox*, expert testimony that replaces a missing, claim-specific showing with generalized “must be present” logic is entitled to little weight. *Xerox*, IPR2022-00624, Paper 9 at 15–17.

5. The expert declaration should be afforded little weight under *Xerox*

These examples demonstrate a repeating pattern. When the Petition's mapping requires proof of a claim-specific mechanism, the Expert Declaration supplies it through labels, conclusory "operates the same way" statements, and generalized inevitability reasoning rather than through a reasoned technical explanation tied to Birse's actual disclosures. Petition, 47–48 and 52–53; EX1003, ¶¶256, 264. Under *Xerox*, that kind of testimony should be afforded little weight and should not be credited to carry Petitioner's burden at institution. *Xerox*, IPR2022-00624, Paper 9 at 15–17.

VII. PETITIONER HAS NOT MET ITS BURDEN TO ESTABLISH OBVIOUSNESS UNDER 35 U.S.C §103

In Ground 1, the Petition alleges that independent claims 1 and 15 of the '282 Patent are obvious over Birse alone. As explained below, Ground 1 is deficient because the Petition has failed to explain how all of the elements of independent claims 1 and 15 of the '282 Patent are met by Birse.

A. The Petition Fails To Explain How Birse Discloses A Union Block Device That Comprises A Low-Level Driver For Interfacing Between The First And Second Storage Units And The File System Of The Compute Node

Claim 1 recites a union block device for "interfacing between the first and second storage units and the file system of the compute node." By way of example,

this claim limitation corresponds to the disclosure in the specification where the file system "reside[s] on each of the compute nodes 120a-n," and the "low-level driver determin[es] which data blocks are needed for the compute node's instance of the application environment and deliver[s] the application environment to the compute [node] via the compute node's file system." EX1001, 4:46-47 and 6:18-22 (emphasis added). Moreover, as explained in Section III above, the file history demonstrates that a union block device broadly described as simply "interfacing between the compute node and the first and second storage units" was insufficient from the Examiner's perspective to distinguish claim 1 from the prior art. Rather, as a condition to allowing the Application, the Examiner required that the language of claim 1 go further and specify a union block device for "interfacing between the first and second storage units and the file system of the compute node." EX1002, 90 and 112.

1. The Petition's Birse mapping proves redirection to server-side storage, not interfacing between two storage units and the compute node's file system

The Petition identifies the limitation in claim 1 requiring that "the union block comprises a low-level driver for interfacing between the first and second storage units and the file system of the compute node," as Element [1.6], and devotes three sentences to explaining how this element is met by Birse. The totality of the Petition's analysis for this claim element is as follows:

As described above, Birse teaches the use of a block device **driver** that “keep[s] track of which portions of the operating system ... have been modified by the user” and redirects file system read and write requests between the system volumes and shadow volumes as appropriate. Ex.1005, 12:11-15. See FIG. 1 of Birse:

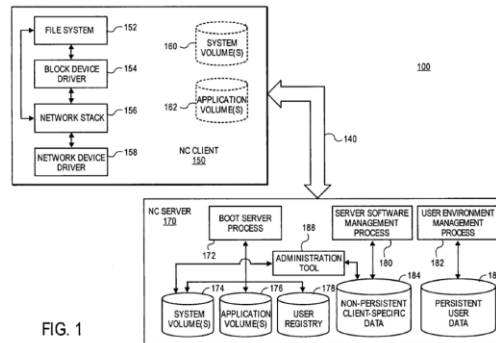


FIG. 1

The “block device driver 154 appears to the file system 152 [of NC client 150] to be a standard hard drive device driver.” *Id.*, 7:35–38. Additionally, Birse specifies that the file system and block device driver operates “as if the contents of the system volumes 160 are, in fact, contained on a local hard drive and the file systems read and write requests are redirected at a **lower level** of the operating system.” *Id.*, 7:27-31. Ex.1003, ¶¶261-63.

Petition, 50-51 (emphasis added).

The Petition's analysis of Element [1.6] fails to explain how Birse's block device 154 comprises "a low-level driver for interfacing between the first and second storage units and the file system of the compute node," as required by the claim language. Among other deficiencies, the Petition fails to explain how the use of Birse's block device 154 to redirect file system read and write requests equates with "interfacing between the first and second storage units and the file system of the compute node," as claimed. Redirecting file system read and write requests is an operation that avoids interaction with the compute node's file system by routing

communications around it -- which is exactly the opposite of what the claim requires. Rather, claim 1 requires, *inter alia*, that the union block device performs interfacing between the first and second storage units and the file system of the compute node." At the very least, "interfacing" requires communicating with the file system of the compute node as opposed to routing communications around the file system of the compute node as set forth in the Petition's mapping of Element [1.6].

Claim 1 requires a low-level driver "for interfacing between the first and second storage units and the file system of the compute node." EX1001, 7:29-33. Petitioner's mapping does not do the necessary architectural work to satisfy that limitation. Instead, Petitioner simply equates the claimed "interfacing" with Birse's statement that the file system's requests are "redirected at a lower level." Petition, 50-51; EX1005, 7:27-31.

2. Petitioner never identifies, in Birse, the claim-required "first storage unit" and "second storage unit" that the low-level driver interfaces with on the compute node side of the file system

The Petition's three-sentence analysis of Element [1.6] also fails to explain how Birse's redirection layer actually interfaces with two storage units in a root-image and leaf-image architecture and delivers their merged result to the compute node's file system as explicitly required. EX1001, 7:22-33. Put differently, Petitioner relies on the fact that Birse has a block device driver, but it never shows

that Birse’s block device driver is structurally positioned and functionally configured as the claimed “union block device” that sits between the compute node’s file system and the first and second storage units that store the root and leaf images. EX1001, 7:22-33; Petition, 50-51.

3. The Petition is defective because it fails to map Birse to Element [1.6]

At institution, Petitioner must demonstrate a reasonable likelihood of prevailing with a coherent, claim-specific mapping. Here, Petitioners' three-sentence analysis with respect to Element [1.6] supports only the proposition that Birse uses a block device driver to make remote system volumes appear local by redirecting read and write requests at a lower level of the operating system. Petition, 50-51; EX1005, 7:27-31 and 7:35-38. Petitioners do not explain how that redirection architecture satisfies the specific claim requirement that the union block device comprises a low-level driver for interfacing between the first and second storage units and the file system of the compute node. EX1001, 7:22-33. Because the Petition never establishes that required topology in Birse, Ground 1 fails to carry Petitioner’s burden for at least element [1.6], and institution should be denied on that basis.

B. The Petition Fails To Explain How Birse’s User Specific “Shadow” Copies Are the Claimed Node-Specific “Leaf Images”

The Petition maps Birse’s “shadow” volumes/image to the claimed “leaf images” in independent claims 1 and 15. Petition, 47-48 and 52-53. That mapping fails because the challenged claims, as interpreted by Southwest in the district court and read in light of the specification, require a leaf image that is a node-specific image of the application environment," meaning that the leaf image stores node-specific block deltas against a shared root image. EX2008, 10-12; EX1005, 2:62–3:3 and 9:2–10. By contrast, Birse’s “shadow” volume or image construct is expressly user-specific and is designed to preserve and roam user preferences across different NC clients (compute nodes). EX1005, 2:62–3:3, 6:35–48 and 9:2–10.

1. According To Petitioner Southwest, the ’282 Patent’s “leaf image” is compute-node- specific and captures node specific block deltas

As Southwest argued in the district court, the ’282 Patent describes a root and leaf architecture in which a shared root image provides a baseline, while changes are captured in a leaf image that is unique to the compute node. Ex.2008, 10-12. Thus, according to Petitioner Southwest, the claimed leaf image, is not merely any repository of modifications. It is the node’s own block level delta image for that node’s particular instantiation of the application environment.

2. Birse’s “shadow” volumes are, by contrast, user specific and intended to roam across clients

Birse is centered on preserving user customizations and delivering a consistent user experience regardless of which NC client the user logs into. Birse teaches preserving “user customizations, such as preferences,” by maintaining “individual, user, storage areas,” so that the user may login “to any NC client on the network” and have the same user experience. EX1005, 2:59 to 3:3.

Birse’s user environment management process likewise stores and retrieves user information from a user specific storage location on the server at session termination, and later retrieves that same user specific data at the next login, whether from the same or a different NC client. *Id.*, 6:35-48.

Birse also characterizes its shadow image in user centric terms, stating that the shadow image preferably contains only portions of the shared operating system image that have been modified by the user. *Id.*, 12:59-61. Birse similarly describes the shadowing function in terms of user preferences, using “if a user modifies a preference” as an exemplar. *Id.*, 14:49-53 and 14:61-67.

In short, Birse’s shadow architecture is designed to preserve and roam user-level personalization, not to create and maintain a node-specific delta image tied to a particular compute node’s instance of an application environment.

3. The Petition's mapping improperly collapses user roaming overlays into node-specific leaf images

The Petition maps Birse's shadow volumes to the claimed leaf images by relying on Birse's statement that a user's preferences are dynamically merged with the system environment provided to the NC client. Petition, 47-48, citing EX1005, 9:2 to 9:5 and, 14:61 to 14:67; and EX1003, ¶¶ 255-256). The Petition likewise relies on Birse's redirection of writes into the user's shadow volume on the NC server and then labels that user shadow volume as the claimed leaf image. Petition, 52-53, citing EX1005, 7:35-53, 10:63-65 and 11:6-17; and EX1003, ¶¶ 265-266).

That mapping does not satisfy the claim requirement as Southwest argued in the district court. Ex.2008, 10-12. As Southwest argued in the district court, the '282 Patent's leaf image is unique to the individual compute node, meaning that it captures changes made by the respective compute node to its instance of the application environment regardless of which user is logged into the node. *Id.* Birse, by contrast, is expressly engineered to do the opposite: make it so that a user can login to any NC client and receive the same user experience because the modifications being preserved are user specific and are retrieved at login regardless of which client (compute node) is being used by the user. EX1005, 2:59-3:3 and 6:35-48.

Petitioners never explain why a user roaming preference overlay, which Birse defines in terms of individual user storage areas and cross client portability, should

be treated as a compute-node-specific leaf image that stores node specific block deltas to a node's instance of the application environment. Petition, 47-48 and 52-53. This conceptual mismatch is not mere semantics. It is a technical incompatibility between Birse's user portability feature, and the node-specific delta images that Southwest argues are required by the claims of the '282 Patent. Ex.2008, 10-12.

4. Petitioners' Expert Confirms the Deficiency in the Petition's Attempted Mapping Of Birse To The Claimed "Leaf Images"

The deficiency in the mapping of Birse to the claimed "leaf image" is also visible in the Petition's reliance on Dr. Zadok. Dr. Zadok quotes Birse's disclosure that user preferences are dynamically merged with the system environment and that the user may login to any client and have the same user experience. EX1003, ¶¶258–259, quoting EX1005, 9:2–10. Yet, for the "leaf image" mapping, Petitioners and their expert treat the shadow volumes (which store user-preferences) as the leaf images that the union block device merges with the root image blocks. Petition, 47–48; EX1003, ¶¶255–257. Neither Petitioners nor their expert explain how a user-roaming preference system constitutes a compute-node-specific leaf image that "describe[s] the changes made by the respective compute node" to its instance of the application environment, as argued by Southwest in the district court. This failure to reconcile the user-centric "shadow" concept with the node-centric "leaf image" requirement advanced by Southwest in the district court, is an independent reason

Ground 1 fails to establish a reasonable likelihood of prevailing.

C. Petitioner’s Inconsistent Identification of the Claimed “Application Environment” Undermines Its Ground 1 Mapping

Ground 1 also fails because Petitioner never maintains a consistent, claim-faithful identification of what the challenged claims call “the application environment.” The ’282 Patent uses that term to describe a specific, claim-required output: a cohesive image presented to a compute node that is created by a union block device through block-level merging of a shared root image and a leaf image. EX1001, 2:3–5; EX1001, 6:10–16; EX1001, 7:24–33. Petitioner’s mapping to Birse does not track that construct. Instead, Petitioner alternates between two materially different concepts, treating the “application environment” as a stored operating system that reflects “a standard useable state” for Elements [1.0]/[15.0], while recasting the “application environment” as a “dynamically merged system environment” that includes user preferences for Elements [1.5]/[15.4]. Petition, 26-27, 47-48.

Starting with Elements [1.0]/[15.0], the Petition's mapping of the independent claims begins by mapping the "application environment" to the "stored copy of the system" used to boot an NC client. There, the Petition (at 26-27) states:

Birse discloses a system and method for distributing an application environment (from NC server 170) to a plurality of compute nodes (NC clients 150, 812, 814, 816, 822, 824, 826): “[T]he NC clients [150] boot over the network by accessing a stored copy of the

operating system from the NC server [170]. NC server.” Ex.1005, 4:56-60. (emphasis added).

Birse explains that the "stored copy of the system" used to boot an NC client corresponds to an operating system in "a standard, useable state":

The NC system environment and the net-booting process described herein are intended to provide the needed reliability. ... [F]or administrative and maintenance purposes, preferably only the NC server 320 has a copy of the master read-only operating system image and NC clients boot according to the net-booting process described below. Since rebooting an NC client restores the NC client to a standard, useable state, it is impossible for a user without proper access privileges to make an NC client unbootable. Consequently, this NC system architecture and net-booting approach greatly simplifies NC client administration and provides a high level of reliability for the NC clients. EX1005, 9:32-50 (emphasis added).

However, in connection with Elements [1.5]/[15.4], the Petition pivots from mapping the "application environment" to "a standard, useable state," to mapping the "application environment" to a customized environment that includes "the user's preferences." Petition, 47-48.

The expert declaration repeats and compounds the same slippage by equating the “application environment” with “stored operating system volumes” (standard, useable state) in one portion of the analysis while defining the same "application environment" as including the user-preference overlay onto a “system environment” in another portion. EX1003, ¶¶182, 217, 256–257.

These internal inconsistencies are not semantic quibbles. Because the claims require that the union block device “creates the application environment by

merging” root-image and leaf-image blocks via a low-level driver beneath the file system, Petitioner must identify a single Birse counterpart that is both the distributed environment and the merged, created output. EX1001, 7:24–33. Petitioner does not do so. As a result, Ground 1 rests on shifting constructions and fails to provide the coherent, evidence-based, element-by-element showing required to demonstrate a reasonable likelihood of prevailing.

1. The intrinsic record ties “application environment” to a specific claim-required output created by block-level root/leaf merging

The challenged claims do not use “application environment” as a free-floating label. Claim 1 requires a “union block device” that “creates the application environment by merging the blocks of the root image stored on the first storage unit with the blocks of the leaf image stored on the second storage unit,” and further requires that the union block device “comprises a low-level driver” interfacing between the storage units and the file system. EX1001, 7:24-33. The specification reinforces that “creating the application environment” is accomplished by “merging the blocks of the root image with the blocks of the leaf image,” and that the resulting “application environment” will “appear to the compute node as one cohesive image rather than a base image plus its additions, deletions, and modifications.” EX1001, 6:10-16. The specification also describes the compute node’s “own unique and cohesive instance of the application environment.” EX1001, 2:3-5. In short, claim 1

consistently defines the “application environment” as the cohesive, merged (root+leaf) image presented to the node by the union block device, not merely an operating system in the abstract.

2. The Petition maps “application environment” one way for the “distributing” limitation, then a different way for the “creates ... by merging” limitation

In Ground 1, the Petition’s mapping of Elements [1.0]/[15.0] identifies the “application environment” in Birse as the "operating system made available from the server at boot, which Birse explains is the "standard" operating system. For example, Petitioners argue that Birse distributes an “application environment” because “the NC clients [150] boot over the network by accessing a stored copy of the operating system from the NC server [170].” Petition, 26. Dr. Zadok’s declaration adopts the same framing, concluding: “In sum, Birse discloses a system ... for distributing an application environment [Birse using stored operating system volumes].” EX1003, ¶217. Under this mapping, the “application environment” is the stored "standard" operating system obtained from the server during net boot. EX1005, 9:32-50.

But, when Petitioner reaches the substantive “creates ... by merging” limitations in Elements [1.5]/[15.4], Petitioners pivot and now label the same claimed “application environment” as “the application environment (dynamically merged system environment).” Petition, 47-48. In that same discussion, the Petition

relies on Birse’s statement that: “When an NC client boots from the network and accesses a stored copy of the operating system from an NC server, the user’s preferences are dynamically merged with the system environment provided to the NC client.” Petition, 48, quoting EX1005, 2:62-65. That “dynamic merging” statement concerns preferences or customizations being overlaid onto a "standard" system environment.

Accordingly, the Petition uses “application environment” as a moving target. For Elements [1.0]/[15.0], it is the "standard" stored operating system stored on the server. For Elements [1.5]/[15.4], it becomes a “dynamically merged system environment” associated with user preferences.

3. The expert declaration compounds the inconsistency by equating the “application environment” with stored operating system volumes while defining “merging” as preference merging with a “system environment”

The Expert Declaration mirrors the Petition’s internal inconsistency. It characterizes the “application environment” for Elements [1.0]/[15.0] as “stored operating system volumes.” EX1003, ¶217. Then, in addressing Elements [1.5]/[15.4], the declaration asserts that Birse’s “union block devices” are “configured to create said application environments by merging the blocks of said root image ... with the blocks of respective leaf images,” and immediately describes the resulting “application environment” as the “dynamically merged system environment.”

EX1003, ¶¶256-257. That is the same slippage. “Application environment” is treated as the "standard" stored operating system volumes in one part of the mapping, yet the “created” environment is described as a “dynamically merged system environment” grounded in preference overlay language in another part of the mapping. EX1003, ¶¶182, 217 and 257.

4. These inconsistencies are dispositive because Petitioner never identifies a single, claim-matching “application environment” in Birse that is both distributed and created by the recited merging operation

Claim 1’s structure requires Petitioner to identify one consistent mapping for "application environment." The same “application environment” that is being “distributed” must also be what the union block device “creates ... by merging” root-image and leaf-image blocks, via a low-level driver beneath the file system. The claims’ description of that creation step is explicit. Merging root and leaf blocks yields an output that “will appear ... as one cohesive image.” EX1001, 6:10-16. Petitioner’s mapping does not keep those concepts constant.

Nor are the two mappings in any way reconcilable. If the Petition’s mapping of "application environment" in Elements [1.0]/ [15.0] controls, meaning that the “application environment” is the "standard" operating system copy accessed from the server, Petitioner does not (and could not possibly) explain how that "standard" operating system copy is something the union block device “creates ... by merging”

root and leaf image blocks into a cohesive image, as the claims require. The Petition never reconciles its competing identifications. Instead, it proceeds as though “application environment” can refer to whichever “environment” is convenient for the particular limitation being mapped. Because Petitioner does not consistently identify a single Birse counterpart that satisfies both sides of that required relationship, Ground 1 fails to meet Petitioner’s institution burden.

D. Petitioner fails to explain why Birse’s “merging user preferences” is the claimed “merging blocks” of a root image and a leaf image

Independent claim 1 requires that “the union block device creates the application environment by merging the blocks of the root image stored on the first storage unit with the blocks of the leaf image stored on the second storage unit.” EX1001, 7:44–46. This is a storage-architecture limitation. It ties (i) blocks of two images (a root image and a leaf image) to (ii) two storage units, and (iii) requires a union block device that “creates” the application environment by merging those blocks. EX1001, 7:31–49.

Petitioner’s reliance on Birse’s statement that “the user’s preferences are dynamically merged with the system environment provided to the NC client” does not satisfy that burden because Petitioner never explains, at the required level of technical and claim-specific detail, how a preference-level merge is the same

operation as the claimed block-level merge of two images across two storage units. Petition, 48; EX1005, 3:13–16; EX1001, 7:44–46.

1. “User preferences” are not “blocks of a leaf image,” and Petitioner does not bridge that gap

Birse’s cited passage is directed to preserving a user’s desktop experience by combining “preferences” with a “system environment.” EX1005, 3:13–16. That sentence does not identify a root image, does not identify a leaf image, and critically does not identify any act of merging blocks stored in two different storage units. EX1005, 3:13–16.

The claim, by contrast, is explicit. It is the merging of blocks of a root image with blocks of a leaf image that “creates the application environment.” EX1001, 7:44–46. Petitioner’s mapping treats Birse’s high-level statement about preferences as if it necessarily describes the claimed block-level image merge, but the Petition does not explain why or how “preferences” correspond to “blocks of the leaf image,” much less how those “preference” blocks are stored on the “second storage unit” and merged with “root image” blocks from the “first storage unit” to “create” the application environment as required. Petition, 48; EX1005, 3:13–16; EX1001, 7:44–46.

2. Birse’s “dynamic merge” can be accomplished at a semantic or user-profile level, not a block-image level

Even taking Birse at face value, “dynamically merged” in Birse describes a functional result, meaning user customizations are some how applied so the user experiences the same environment across clients. EX1005, 3:13–16. That result can be achieved through many mechanisms, for example copying or applying configuration data at login or session initialization, without any need to merge the blocks of two images stored on two storage units.

Because the claims require a specific mechanism, namely block merging of root-image blocks and leaf-image blocks, Petitioner must do more than quote Birse’s functional statement. Petition, 48; EX1005, 3:13–16; EX1001, 7:44–46. Petitioner must articulate why Birse’s “dynamic merge” is, in fact, the claimed block-level merge performed by a union block device between two storage units. EX1001, 7:44–46. The Petition never supplies that missing explanation. Petition, 48–49.

3. The Petition’s own discussion shows redirection or selection, not “merging blocks ... to create the application environment”

The Petition’s mapping elsewhere emphasizes that Birse’s block device driver decides whether a requested portion “has been modified” and then directs reads to the system volumes or the shadow volume. Petition, 49–51; EX1005, 11:39–41, 11:55–12:10. That is request-routing or copy-on-write overlay behavior. Petition,

49–51; EX1005, 11:39–41, 11:55–12:10. It is not, without considerable further explanation, the same thing as a union block device that “creates the application environment by merging” blocks of a root image with blocks of a leaf image. EX1001, 7:44–46.

Put differently, the Petition appears to treat “the user’s preferences are dynamically merged with the system environment” as a synonym for the claim’s “merging blocks” requirement, but it never explains what exact blocks are being merged in Birse, where Birse stores the root-image blocks and leaf-image blocks on respective first and second storage units, and how that merge “creates the application environment,” as opposed to merely resolving individual read and write requests during operation. Petition, 48–51; EX1005, 3:13–16, 11:39–41 and 11:55–12:10; EX1001, 7:44–46.

4. The Board should reject Petitioner’s conclusory equivalence between “preference merge” and “block merge”

Because the Petition’s obviousness case depends on establishing that Birse meets the specific limitation of “merging the blocks of the root image ... with the blocks of the leaf image ...,” Petitioner must provide a coherent, technically grounded mapping that ties Birse’s disclosure to that claim language. EX1001, 7:44–46. The Petition instead relies on an unelaborated functional statement about user preferences and treats it as dispositive of the claimed block-level image merge.

Petition, 48; EX1005, 3:13–16. That is insufficient to meet Petitioner’s burden, particularly where the claim requires a specific storage, image, and block mechanism rather than merely the general outcome of “user customization.” EX1001, 7:44–46.

E. The Additional Grounds All Depend On The Deficient Mapping Of Birse To The Independent Claims In Ground 1

Grounds 2 through 4 do not present alternative mappings for the core limitations of independent claims 1 and 15. Instead, Petitioners repeatedly premise each additional ground on the assertion that “Birse meets all elements of independent Claims 1 and 15,” and then cite the secondary references only for collateral, claim-specific add-ons (HPC or grid context, a modification log, or partitioning and snapshot merging). Petition, 76, 77, 82 and 85. Accordingly, since the Petition fails to establish that Birse teaches the requirements of independent claims 1 and 15 discussed above, the additional grounds necessarily fail as well.

1. Ground 2 (Birse + Ballew): Ballew Adds Only HPC or Grid Context and Does Not Address the Deficiencies In Ground 1 With Respect To The Independent Claims

In Ground 2, Petitioners expressly rely on their Ground 1 contention that Birse already satisfies all limitations of claims 1 and 15. Petition, 76-77. Ballew is then introduced only for the dependent limitations requiring that the compute node

“comprises a server” and that the system or method “operates in a high performance computing cluster” or “grid computing cluster.” Petition, 76-79.

The Petition’s motivation-to-combine discussion confirms this narrow role. Petitioners state that Ballew “discloses providing an execution environment generally to nodes in an HPC system,” while “Birse offers details on OS storage and distribution,” and therefore the operating system of Ballew’s execution environment would be distributed “using Birse’s shadow volumes.” Petition, 75-76. That explanation does not purport to supply any missing elements identified in Ground 1. It assumes Birse already satisfied all of the elements of the independent claims. Critically, Ballew is not cited to resolve the specific deficiencies that undermine Ground 1. Because Ballew is not used to cure any of the asserted shortcomings of Birse, Ground 2 rises or falls with Ground 1.

2. Ground 3 (Birse + Wu): Wu Is Offered Only for a “Block Modification Log,” and the Petition Concedes the Key Architecture Still Depends on Birse

Ground 3 makes the dependency on Birse even more explicit. Petitioners state, “To the extent Birse does not explicitly disclose a block modification log, Wu discloses a block modification log,” and then conclude that “Birse as combined with Wu meets Claims 8 and 24.” Petition, 82. That is the only purported contribution Wu makes.

Wu is not cited to address any of the deficiencies of the Petition in mapping the independent claims to Birse in Ground 1, as discussed above. Petitioners do not rely on Wu to define or consistently identify an “application environment,” to show a union block device that creates that environment by merging blocks of a root image and a leaf image, to establish a leaf image, or to satisfy the claim-required low-level driver topology. Instead, the Petition’s Ground 3 mapping again assumes Birse supplies the complete independent-claim framework and uses Wu only as an add-on for the “block modification log” limitation. Petition, 81-82. Because Wu is not offered to cure the deficiencies with respect to mapping Birse to the independent claims, Ground 3 fails for the same reasons as Ground 1.

3. Ground 4 (Birse + Kleinschnitz): Kleinschnitz Is Offered Only for Snapshot Collapsing and Partitioning, Not for the Missing Birse Elements

Ground 4 similarly confirms that Kleinschnitz is not offered to cure any of the shortcomings of Ground 1 with respect to the independent claims. Petitioners’ motivation-to-combine section focuses on two concepts: (1) “reconciling” as “nothing more than merging snapshots,” and (2) storing system volumes and shadow volumes in different partitions “like in Kleinschnitz.” Petition, 84. The element-by-element section then cites Kleinschnitz only for partitioning (claims 9 and 25) and incremental-image merging (claim 17). Petition, 85-86.

Nothing in Ground 4 suggests Kleinschnitz is being used to supply the missing proof for the requirements of the independent claims. Instead, Ground 4 proceeds on the premise that Birse already meets the independent claims and then layers on generic snapshot-collapsing and partitioning concepts from Kleinschnitz. Petition, 84-86. For institution purposes, this is dispositive. Ground 4 does not offer an alternative path to meeting the limitations of the independent claims.

4. Grounds 2–4 Do Not Independently Establish Unpatentability

Grounds 2 through 4 do not provide an independent path to establishing unpatentability because Petitioners do not cite Ballew, Wu, or Kleinschnitz to cure the deficiencies in Birse’s disclosure of the independent-claim architecture. The Petition’s own structure confirms this. Ground 1 is asserted “over Birse” and begins with the premise that “Birse meets all claim elements.” Petition, 26. Grounds 2 through 4 are then presented as combinations, each with a stand-alone “Motivation to Combine” section and a “Birse and [secondary reference] meet all claim elements” section. Petition, 74–86. In each of those grounds, the secondary reference is used only for discrete dependent-claim features (cluster context, a modification log, partitioning/snapshot merging), while the Petition continues to rely on Birse for the core limitations of independent claims 1 and 15, including the union block device and its low-level driver topology, the node-specific leaf image, and the claimed

block-level creation of the application environment. See, e.g., Petition, 74–86; EX1001, 7:24–33.

VIII. CONCLUSION

Petitioner has not carried its burden to establish a reasonable likelihood that any challenged claim of the '282 Patent would have been obvious under 35 U.S.C. § 103 because the Petition fails to demonstrate that several limitations of the independent claims are met by Birse. For example, the Petition fails to demonstrate that Birse discloses a union block device comprising "a low-level driver for interfacing between the first and second storage units and the file system of the compute node," as required by independent claim 1.

In addition, Petitioner Southwest is advancing inconsistent claim constructions here and in the district court with respect to the term "leaf images" in independent claims 1 and 15 of the '282 Patent. While in these proceedings Southwest interprets "leaf image" according to its "ordinary and customary meaning," in the district court, Petitioner Southwest opposes interpreting "leaf images" according to its plain meaning and asserts that "a 'leaf image' is a node-specific image of the application environment operating at the block level below the file system." Ex. 2008, 12 (emphasis in original). The Petition fails to demonstrate

that Birse discloses a node-specific leaf image, as required by Petitioner Southwest's district court claim construction.

Petitioner American is also advancing inconsistent claim construction positions. While American's expert in these proceedings was able to interpret the claims according to their "ordinary and customary meaning," American contends in the district court that the phrases "appropriate persistent mapping" (in independent claims 1 and 15), "the modifying" (in claim 15) and "at an operational level" (in claim 26) are all indefinite. EX2009, 4-5. As Director Squires explained in *Tesla, Inc. v. Intellectual Ventures II LLC*, IPR2025-00340, Paper 18 (November 5, 2025) (informative), absent a sufficient explanation why inconsistent claim construction positions are warranted, a petitioner cannot not argue that a claim is indefinite in the district court and at the same time maintain that the same claim should be interpreted according to its plain and ordinary meaning in IPR proceedings. Here, the Petitioners fail to even acknowledge their existence of inconsistent claim construction positions, yet alone explain why inconsistent claim construction positions are warranted. This deficiency provides an independent basis for denial of institution due to the Petition's failure to satisfy its burden to explain how the challenged claims are to be construed under 37 CFR42.104(b)(3).

The Petition's allegations that Birse discloses the "application environment" recited in the independent claims of the '282 Patent are also defective. With respect

to that limitation, the mappings in the Petition are inconsistent and rely on shifting identifications of key claim concepts and conclusory expert assertions instead of record evidence. For example, Petitioner inconsistently characterizes the “application environment” in Birse, alternately treating it as a stored operating-system copy distributed from a server and as a “dynamically merged system environment” based on user preferences, without reconciling those positions or mapping either of them to the claimed cohesive, block-merged image. Petition, 18, 26, 47–53; EX1001, 6:10–16; EX1003, ¶¶217, 255–266. And Petitioner does not establish the claim-required interface topology for the “low-level driver,” instead pointing to Birse’s generic statement that requests are “redirected at a lower level of the operating system,” which is not the same as a driver that interfaces between two image storage units and the compute node’s file system as claimed.

Finally, Grounds 2 through 4 do not provide an independent path to unpatentability. Petitioners do not cite Ballew, Wu, or Kleinschnitz to cure any deficiency in Birse’s disclosure of the independent-claim architecture and instead rely on those references only for discrete dependent-claim features. Petition, 74–86. Because the Petition’s threshold showing on Birse fails, the additional grounds necessarily fail as well.

For the foregoing reasons, Patent Owner respectfully requests that the Board deny institution of inter partes review.

Respectfully submitted,

Dated: January 20, 2026

By: /Brandon R. Theiss/
Brandon R. Theiss
Registration No. 70,507
Counsel for Patent Owner
Intellectual Ventures I LLC

CERTIFICATE OF COMPLIANCE WITH WORD COUNT

Pursuant to 37 C.F.R. § 42.24(d), I certify that this Patent Owner Preliminary Response complies with the type-volume limits of 37 C.F.R. § 42.24(b)(1) because it contains 11,638 words, excluding the parts of this Patent Owner's Preliminary Response that are exempted by 37 C.F.R. § 42.24(a), according to the word processing system used to prepare this Patent Owner's Preliminary Response.

Respectfully submitted,

Dated: January 20, 2026

By: /Brandon R. Theiss/
Brandon R. Theiss
Registration No. 70,507
Counsel for Patent Owner
Intellectual Ventures I LLC

CERTIFICATE OF SERVICE

The undersigned certifies that pursuant to 37 C.F.R. § 42.6(e), a copy of the foregoing **PATENT OWNER'S PRELIMINARY RESPONSE** was served via email (as consented to by counsel) on January 20, 2026 to lead and backup counsel of record for Petitioners as follows:

John B. Campbell
jcampbell@McKoolSmith.com
MCKOOL SMITH, P.C.
303 Colorado Street, Suite 2100
Austin, TX 78701

Casey Shomaker
cshomaker@mckoolsmith.com
McKool Smith, P.C.
300 Crescent Court, Suite 1200
Dallas, TX 75201

Emily R. Tannenbaum
etannenbaum@mckoolsmith.com
McKool Smith, P.C.
1301 Avenue of the Americas, 32nd Floor
New York, New York 10019

Keith D. Harden
kharden@munckwilson.com
Munck Wilson Mandala, LLP
2000 McKinney Ave., Ste. 1900
Dallas, Texas 75201

S. Wallace Dunwoody
wdunwoody@munckwilson.com
Munck Wilson Mandala, LLP
2000 McKinney Ave., Ste. 1900
Dallas, Texas 75201

Michael C. Wilson
mwilson@munckwilson.com
Munck Wilson Mandala, LLP
2000 McKinney Ave., Ste. 1900
Dallas, Texas 75201

AA_Intellectual_Ventures@mckoolsmith.com

SWA-IV@munckwilson.com

Dated: January 20, 2026

By: /Brandon R. Theiss/
Brandon R. Theiss
Registration No. 70,507
Counsel for Patent Owner
Intellectual Ventures I LLC