

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

GOOGLE LLC,
Petitioner

v.

SANDPIPER CDN, LLC,
Patent Owner

Case IPR2025-01010
U.S. Patent No. 10,057,322

PATENT OWNER'S SUR-REPLY TO PETITIONER'S REPLY

Mail Stop "PATENT BOARD"
Patent Trial and Appeal Board
U.S. Patent and Trademark Office
P.O. Box 1450
Alexandria, VA 22313-1450

TABLE OF CONTENTS

I.	INTRODUCTION	2
II.	CLAIM CONSTRUCTION	2
A.	“direct link”	2
B.	“API”	7
1.	The claimed API must be a defined, distinct layer, separate from the compute engine.	8
2.	Regardless, the claimed API must provide an interface through which an external caller invokes computational functionality.	10
3.	The claimed API recites a defined input-output relationship. .	11
C.	“with a tag”	12
III.	PETITIONER FAILS TO ESTABLISH ANTICIPATION OR OBVIOUSNESS OF ANY CHALLENGED CLAIM.	14
A.	Verma does not teach a “direct link” to a CDN node.	15
B.	Petitioner fails to establish that Verma-Lewin renders obvious the “tag” limitations of independent claim 1.	16
1.	Petitioner’s Combination	16
2.	Petitioner has not established a motivation to combine Verma and Lewin.....	17
3.	Petitioner fails to explain how tag identification triggers retrieval of modified embedded resources.	23
4.	Lewin’s ESI tags do not provide the claimed “embedded resource with a tag.”.....	25
C.	Petitioner fails to establish that the cited art discloses or renders obvious the claimed API.	26
IV.	CONCLUSION.....	29

I. INTRODUCTION

Petitioner's Reply fails to cure the Petition's flawed arguments. The Reply relies heavily on attorney argument, much of which is either conclusory or incorrect, both legally and factually. Petitioner fails to rebut the well-supported opinions of Patent Owner's (PO) expert, Dr. Russ, who explained in detail why none of Petitioner's grounds render the challenged claims obvious. Indeed, Petitioner did not take Dr. Russ's deposition and did not submit any reply declaration countering his opinions with evidence or expert opinion. Petitioner thus fails to engage with PO's evidence at all, much less does Petitioner's Reply overcome it. The Board should uphold the patentability of the claims.

II. CLAIM CONSTRUCTION

A. "direct link"

The POR establishes that the term "direct link" should be construed as a network address that enables a client device to bypass conventional DNS resolution of a hostname when contacting a CDN node. POR, 11-15; EX2015, ¶¶41-46. This is the plain and ordinary meaning of the term in view of the intrinsic record.¹

¹ Patent Owner's construction does not rely on lexicography or disavowal. *See* Reply, 2.

The Specification explains that a “direct link” is one that enables a client device to “directly contact the node.” EX1001, 7:61-65, 8:5-10. It then explains that such *direct* contact is contact in which the client device “bypass[es] DNS and instead *connect[s] directly* to the source of the embedded resource.” EX1001, 5:29-34²; *see also id.*, 2:45-49 (disclosing an “embedded resource providing a connection to a node of a CDN bypassing DNS”), 2:58-62 (same). The specification further explains that DNS is bypassed, for example, “by embedding an IP address to serve the content directly.” EX1001, 5:29-32. Taken together, these disclosures indicate that a “direct link” is one that enables the client to contact the CDN node without performing conventional DNS resolution of a hostname.

The prosecution history does not contradict PO’s understanding—it reinforces it. The “direct link” arguments during prosecution focused on DNS bypassing—not avoiding edge server selection as Petitioner contends, Reply, 5—arguing that the examples in the ’322 patent “allow[] the client to bypass the name server at the CDN when requesting the content at the CDN.” EX1002, 69-70. The prosecution understanding of the term is thus consistent with PO’s current construction.

² Emphasis added unless otherwise noted.

Petitioner argues that the '322 patent does not “narrowly define the form a ‘direct link’ must take,” and that it “could be a hostname, IP address, virtual IP address, or the like.” Reply, 1-2. Even if true, this does not eliminate the term’s DNS-bypassing requirements. POR, 14. As the POR explains, regardless of the form, the direct link must allow conventional DNS processing to be bypassed. POR, 14; EX1001, 5:29-32, 6:60-7:1.

Petitioner also argues that PO’s construction excludes embodiments that teach an “absolute URL” constructed using a “hostname[],” alleging that the specification discloses that such URLs enable client devices to “directly contact the node for the content.” Reply, 4-5 (citing EX1001, 6:60-62). Petitioner ignores, however, that the Specification defines direct contact as “bypass[ing] DNS.” EX1001, 5:29-34; *see also id.*, 2:45-49, 2:58-62. Indeed, the very next sentence after Petitioner’s citation states that implementing a hostname with a “specific reference to the CDN” enables the client to “contact[]” the CDN “directly, bypassing any other DNS processing.” EX1001, 6:60-7:1; *see also id.*, 5:29-32. This is consistent with PO’s construction.

Petitioner’s claim differentiation argument likewise fails. Petitioner argues that, because claim 14 recites different components that may be included in the modified embedded resource (e.g., IP address, Virtual IP address, absolute URL), the definition of “direct link” must be broad enough to encompass these. *See*

Reply, 3-4. Not so. Claim 14 does not recite or otherwise limit the direct link. That a *modified embedded resource* may include various types of information does not impact the definition of the *direct link*. Moreover, the Specification distinguishes between a “direct link” and a “virtual IP address,” confirming that not all addressing mechanisms qualify as a “direct link.” EX1001, 7:61-65.

Petitioner further argues DNS bypassing is not required because, “at most,” the claims “offer an alternative to using DNS to select a node,” and “the invention as a whole” does not “avoid[] all DNS processing.” Reply, 5-6 (emphasis omitted). This argument misses the mark. While DNS-based selection avoidance may be *one* objective of the claims, this does not affect the meaning of a “direct link.” The intrinsic record makes clear that a “direct link” is one that allows the client device to contact the CDN node without performing conventional DNS resolution of a hostname. That the system may *also* avoid DNS-based server selection does not alter that meaning, nor does the system’s use of DNS processing elsewhere (e.g., for resolving the origin server address, EX1001, 4:5-15).

Petitioner finally argues that PO’s construction contradicts its infringement contentions in district court. Reply, 2-3. No so. The contentions merely provide examples of links that are believed to qualify as direct links. EX1019, 7 (providing an example (“e.g.”) of a direct link “pointing to [a] specific cache server”). They

do not redefine the term or suggest that URLs requiring conventional DNS resolution would satisfy the claim. Petitioner identifies no inconsistency.

For the above reasons, the Board should adopt PO's plain meaning construction. The Board should also reject Petitioner's contrary construction—"a link pointing to a specific server." Reply, 2. Petitioner's construction conflates the destination (e.g., a particular server) with the path to reach that destination (e.g., a direct connection). That a link specifies a particular server says nothing of the route taken by the network to reach that server.

Petitioner purports to justify its construction because "[t]he claim language does not narrowly define the form a 'direct link' must take" or "prohibit all DNS processing." Reply, 1-2. These arguments fail for the reasons provided above. Indeed, to the extent other forms are possible, this does not eliminate the requirement that the link must enable client devices to bypass conventional DNS resolution.

Further still, Petitioner's construction contradicts its own arguments against PO's construction. Petitioner argues that PO's construction "reads out other embodiments, like a hostname," Reply, 4, but it is *Petitioner's* construction that actually does this. The hostname-based URLs described in the Specification do not point to a specific server. Rather, these URLs are directed to the CDN "name servers" so that the CDN may "select[]" the server to be returned. EX1001, 6:60-

7:1. Petitioner argues such URLs constitute direct links, but its construction excludes them.

Accordingly, in view of the '322 patent, the proper interpretation of “direct link” is a network address that enables a client device to bypass conventional DNS resolution of a hostname when contacting a CDN node. EX2015, ¶46.

B. “API”

As PO’s expert explains, the claimed “API” is a defined programmatic interface that specifies the inputs a calling system must supply and the output the receiving system will return. EX2015, ¶106; POR, 40-46, 50-51.³ Petitioner does not offer a contrary definition. It argues only that the claimed API may include the functionality of the compute engine and that it does not require a specific input-output relationship. *See* Reply, 7-11. Petitioner is wrong. The API must be a defined, distinct interface layer, separate from the functionality of the compute engine. Even if integrated with the compute engine, the API must provide a defined interface layer apart from the compute engine’s computational

³ Contrary to Petitioner’s assertion, this is the only interpretation Patent Owner has offered for this term. Dr. Russ’s previous general characterizations of APIs were explanatory—not intended to define the metes and bounds of the term. EX2015, ¶105.

functionality—that is, the API must still function as an interface having a defined input-output relationship through which an external caller invokes computational functionality.

1. The claimed API must be a defined, distinct layer, separate from the compute engine.

The claimed API is a defined, distinct layer, separate from the compute engine. POR, 40-44. The Specification’s statement that “the API may include the functionality of the engine and not necessarily be considered a discrete mechanism” describes an alternative embodiment, not the architecture recited by the claims. POR, 41-44; EX1001, 7:20-25.

The claims recite the API receives inputs “from an origin system that is separate from the content delivery network.” EX1001, 10:63-67. The API thus acts as an interface for an external system to communicate with the CDN. The alternative embodiment, conversely, “does not necessarily involve a customer origin server or other devices outside the CDN.” EX1001, 7:10-15; POR, 40-44. In this embodiment, the CDN itself performs the “process of contacting the API,” eliminating any need to expose an API to an external origin system. EX1001, 7:10-15. This alternative embodiment thus allows the API to be integrated with the compute engine because, unlike the claimed embodiment, no external interface function is required.

Petitioner admits that this embodiment is an “alternative implementation,” yet maintains that the claimed API “can comprise the engine.” Reply, 8. Petitioner ignores that “a claim need not cover all embodiments,” and that a “patentee may draft different claims to cover different embodiments.” *Intamin Ltd. v. Magnetar Techs., Corp.*, 483 F.3d 1328, 1336-37 (Fed. Cir. 2007). Here, the claims encompass the claimed embodiment—not the alternative implementation lacking an external interface.

Petitioner’s reliance on *Liebel-Flarsheim Co. v. Medrad, Inc.*, 358 F.3d 898, 913 (Fed. Cir. 2004) is misplaced. That case addresses situations where the “written description of the invention is narrow, but the claim language is sufficiently broad that it can be read to encompass features not described in the written description.” *Liebel-Flarsheim*, 358 F.3d at 905. That is not at issue here, where the written description describes multiple embodiments, only one of which is claimed.

Petitioner further argues that claims 11 and 15 encompass the integrated API/compute engine embodiment because they “flexibly leave open where generating the modified embedded resource may occur (i.e., at the API or elsewhere in the CDN).” Reply, 9. Petitioner is wrong—this flexibility is indicative of the claimed embodiment. The claims recite the API receives an embedded resource “from an origin system that is separate from the content delivery

network,” but do not recite the API generates the modified embedded resource.

EX1001, 10:64-67. This is consistent with the claimed embodiment in which the API receives the inputs from an external origin system and the compute engine generates the output. The claims recite that the API is an interface accessible to an external caller (e.g., the “origin system”). If the claims contemplated the alternative embodiment—where the API and the engine are collapsed into a single internal CDN component—there would be no reason to recite a separate API at all.

A POSA would have thus understood the claimed API to be the interface disclosed in the primary embodiment described in the Specification—a defined, external-facing layer through which the origin system invokes CDN functionality. *See* POR, 41-44.

2. Regardless, the claimed API must provide an interface through which an external caller invokes computational functionality.

Even if the claims permitted the API to be integrated with the underlying compute engine, the requirement of providing a defined API still exists and should not be read out of the claims. POR, 44-46. In other words, the API must still provide a defined interface through which an external caller invokes computational functionality. The alternative embodiment provides only that the API may be architecturally integrated with the engine’s functionality—it does not remove the requirement that an API be provided.

The Specification confirms this functional requirement. It distinguishes API-based communication from other forms of communication that do not use an API. For instance, it distinguishes “calling an API” from “*otherwise querying* the CDN,” and also discloses “*other* forms of communicating with ... the computing engine” that do not involve an API. EX1001, 3:9-14, 7:20-22. This distinction has meaning only if the API denotes a particular type of communication—i.e., through a defined interface.

Contrary to Petitioner’s assertion, *see* Reply, 10-11, the claims reinforce this interpretation. Claim 1 recites the CDN to provide a modified embedded resource to the origin system, but does not recite an API. Conversely, claims 11 and 15 explicitly recite communication through an API. Consistent with the Specification, claim 1 thus encompasses broader forms of communication between the origin server and content delivery network than claims 11 and 15.

3. The claimed API recites a defined input-output relationship.

As explained, the claimed API denotes a defined programmatic interface that specifies the inputs a calling system must supply and the output the receiving system will return. EX2015, ¶106; POR, 50-51. Petitioner argues that the API need not have “a specific interface” with a “specified input-output contract” because the Specification allows flexibility in the API’s inputs and outputs. Reply, 9-11. According to Petitioner, the Specification describes various inputs to the API, such

as the “information used in determining the [user] address,” and various output forms of the API, such as “an absolute URL ..., and/or a list of hostnames and/or IP addresses.” Reply, 9-10. This argument fails.

That the Specification describes flexibility in what inputs and outputs may be used does not eliminate the requirement that the API implement a defined input-output structure. Indeed, PO’s construction does not allege a *particular or specific* input-output structure—it merely states that there be one. Without such structure, there is no distinction between API-based communication and merely forwarding a request through an internal CDN pipeline. EX2015, ¶¶106-108.

C. “with a tag”

The POR establishes that the plain and ordinary meaning of an “embedded resource with a tag” is an embedded resource that *includes* a tag. POR, 28-31. Petitioner argues that this term merely requires an embedded resource to have a tag “identifying, associated with, or included in it.” Reply, 6-7. Petitioner’s proposal is impermissibly broad and untethered from the Specification, which consistently uses “with” to denote inclusion—not mere association.

The Specification interchangeably describes embedded resources “with” a tag and embedded resources that “include” a tag to mean the same thing—tags that are included within, or part of, the corresponding embedded resource. EX1001, 1:57-62, 4:32-34; POR, 29. A POSA would have thus understood the claimed

“embedded resource with a tag” to mean an embedded resource having a tag included therein. *Edwards Lifesciences LLC v. Cook Inc.*, 582 F.3d 1322, 1329 (Fed. Cir. 2009) (“The interchangeable use of the two terms is akin to a definition equating the two.”).

This understanding is reinforced by the Specification’s consistent use of “with” to denote inclusion. For instance, the Specification explains that modified embedded resources are “directly embedded” into HTML documents. EX1001, 6:53-56; *see also id.*, 1:13-18. It then *interchangeably* characterizes these HTML documents as ones that “*include* a modified embedded resource” and ones “*with* the modified embedded resource.” EX1001, Abstract, 1:57-63, 2:14-17, 4:42-48, 7:48-55, 5:29-34; *see also id.*, 2:45-62, 4:22-28, 4:56-60. Similarly, the Specification *interchangeably* describes modified embedded resources as ones that “*include* an absolute URL,” and ones “*with* an address or URL.” EX1001, 4:42-48, 6:56-60. Across these disclosures, “with” consistently denotes that one component is included within another. A modified embedded resource is included within an HTML document, and a URL is included within a modified embedded resource.

In view of this consistent usage, a POSA would have understood the claimed “embedded resource with a tag” to mean an embedded resource that includes a tag. The tag is a part of, or is, the modified embedded resource itself.

Petitioner’s conclusory assertion that the specification “distinguishes” embodiments “with a tag” from embodiments “including a tag” is thus misplaced. Reply, 6. The Specification does not give different meanings to these phrases; it uses them interchangeably to describe the same concept. Petitioner identifies nothing to contradict this. Thus, contrary to Petitioner’s assertion, PO does not “impermissibly read[] a lone embodiment into the claims.” Reply, 6. It captures the plain and ordinary meaning of the term in view of the Specification.

Petitioner’s proposed construction is therefore overly broad and untethered from the Specification. It is also contradicted by the claims themselves. When the patentee intended to claim an “associated with” relationship, it did so explicitly—claiming, for instance, “content *associated with* the embedded resource” and a “parameter *associated with* the embedded resource.” See EX1001, 10:5-12:20. Conversely, the patentee chose inclusion language (“with”) to define the relationship between the between the modified embedded resource and the tag. Petitioner’s construction renders this language meaningless.

III. PETITIONER FAILS TO ESTABLISH ANTICIPATION OR OBVIOUSNESS OF ANY CHALLENGED CLAIM.

The Petition fails to establish that Verma, alone or in combination with Raciborski, Lewin, or Draai, anticipates or renders obvious any challenged claim of the ’322 patent. Petitioner’s Reply does not change that.

A. Verma does not teach a “direct link” to a CDN node.

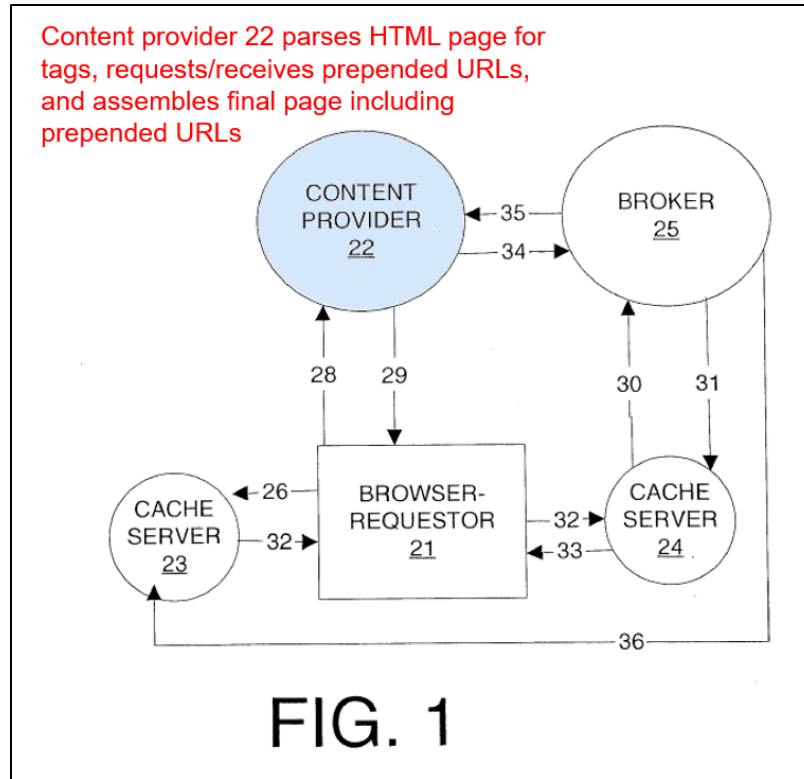
It is undisputed that Verma does not disclose the claimed “direct link” under PO’s construction. *See* POR, 22-25; Reply, 11-13. Petitioner argues only that Verma satisfies this term under its own construction—“a link pointing to a specific server.” Reply, 1, 11 (“No more is required.”). As explained, the Board should reject Petitioner’s construction because it does not find support in the intrinsic record. Consequently, the Board should also find that Verma does not teach the claimed “direct link.”

Verma’s prepended URLs do not contain an IP address and do not otherwise allow DNS-bypassing. EX2015, ¶¶56-58. They require DNS hierarchical processing “to resolve the altered hostname.” EX1009, 9:8-47; EX2016, 55:11-57:8; EX2015, ¶¶58-60. Petitioner argues only that “Verma’s URL precludes DNS from selecting the node.” Reply, 11. It does not argue Verma’s URLs enable client devices to bypass DNS processing, either at the CDN or otherwise. *See* Reply, 11-13; *Section II.A supra*. And for good reason; unlike the “direct link” of the ’322 patent, Verma’s entire routing model depends on conventional DNS resolution of CDN hostnames to direct clients to the appropriate edge server. EX1005, ¶¶12-13, 26-27; EX2015, ¶61. Petitioner thus fails to establish that Verma anticipates or renders obvious a “direct link,” as recited in all independent claims. EX2015, ¶62.

B. Petitioner fails to establish that Verma-Lewin renders obvious the “tag” limitations of independent claim 1.

1. Petitioner’s Combination

Petitioner’s Verma-Lewin combination modifies Verma’s browser 21 to request an “HTML page” from the content provider 22, in view of Lewin. Pet., 44, 48-49. The HTML page includes “an embedded URL” that is “specified by a ‘tag[].’” Pet., 48-49. The content provider 22 accesses and “parses the [requested] HTML page looking for tags,” as taught by Lewin. Pet., 55-56; Reply, 14 (“[I]t would have been obvious to parse Lewin’s tags on Verma’s origin system.”), 20. Upon finding such tags, the content provider requests a “prepended URL” from Verma’s broker 25. Pet., 52, 54-56. The content provider 22 then “assemble[s] a page ... that includes [the] ‘prepended URL[s],’” and returns the assembled page to the browser. Pet., 57, 59; Reply, 20.



EX1005, FIG. 1 (annotated).

2. Petitioner has not established a motivation to combine Verma and Lewin.

Petitioner fails to establish that a POSA would have been motivated to combine Verma and Lewin as proposed. POR, 31-35. Petitioner's theory requires Verma's content provider server to implement Lewin's core functionality—parsing HTML pages for tags and assembling HTML pages—even though Lewin teaches that those functions are performed at CDN edge servers, not by the content provider. As the POR explains, Petitioner fails to explain why a POSA would have relocated these functions to Verma's content provider, and neither reference provides any reason to do so. *See* POR, 31, 34-35.

Verma does not teach HTML parsing or page assembly at all, let alone at the content provider. The content provider is an intermediary between users and the CDN: it receives a content request, forwards the request to the broker, and returns the broker's response identifying a cache server. EX1005, ¶¶12, 25-27, 37-38, FIG. 1. The content provider does not assemble or serve the requested content; the client retrieves the content directly from the cache server. EX1005, ¶¶37-38. Lewin likewise does not support Petitioner's modification. It consistently teaches that HTML parsing and page assembly occur at *CDN edge servers*, not at the origin system. User DNS requests are resolved to the edge server, which parses, assembles, and delivers the page to the user. EX1007, ¶¶49-50.

Despite these disclosures, Petitioner purports to relocate Lewin's HTML parsing and page assembly to Verma's content provider. Yet it does not explain why a POSA would have done so. The entirety of the Petition's explanation for these modifications is as follows:

HTML Parsing: "It would have been obvious for a POSITA to parse an HTML webpage for tags, as in Lewin, to enable Verma's system to utilize and identify advantageous embedded content by providing a solution for dynamic identification and execution of the URLs." Pet., 56.

Page Assembly: "It would have been obvious for Verma's 'content provider' (origin system) to assemble a page ('Movies-Inc.Com' 'web site' 'web

page[]') (content resource) that includes Verma's 'prepended URL' (modified embedded resource) pointing to a cache server storing a requested 'movie,' and return (deliver) that page to Verma's 'client that runs a browser' (client device) for viewing." Pet., 57.

Petitioner's Reply fares no better. There, Petitioner argues that Lewin's tags are not limited to implementation "on an edge server," but identifies no disclosure teaching such an implementation on any other server, let alone an origin content provider. Reply, 19-20. Petitioner asserts that "the relevant inquiry is whether it was obvious to parse and identify tags with embedded resources on Verma's content provider server." Reply, 20. But, again, Petitioner provides no reason as to why it would have been obvious to implement such functionality *on this server*. The Reply likewise fails to provide any reason for implementing page assembly *on this server*. See Reply, 19-21.

That Petitioner structures its combination in this manner is no accident. Independent claim 1 recites the *origin system to obtain a modified embedded resource upon identification of a tag, and to deliver[] the content resource to the client device with the modified embedded resource*. Petitioner maps Verma's content provider 22 to the claimed *origin system*, Lewin's HTML page to the claimed *content resource*, and Lewin's ESI tags to the claimed *tag*. Thus, to satisfy the claim, Verma's content provider 22 must *identify* the ESI tag and *deliver* the

HTML page. Because neither reference discloses those functions at the origin, Petitioner imports Lewin's edge-server functionality into Verma's content provider.

The prior art does not support these modifications. Verma's content provider is merely an intermediary that redirects client requests to cache servers. EX1005, ¶25. It does not assemble or serve content. *See* EX1005, ¶¶37-38. Verma does not provide any teaching or suggestion for doing so, nor does Petitioner point to any.

Lewin similarly provides no basis for implementing such functionality at a content provider. In fact, it teaches the opposite. It teaches that it is "highly desirable" to assemble and serve pages at the edge, from "a location much closer" to requesting users, "*rather than from much farther away at the original Web server.*" EX1007, ¶9. This allows for "faster loading pages, because pages are assembled closer to the end user, *instead of on the origin server*, and reduced traffic/load on the application/web server, because more requests can be satisfied on the network edge." EX1007, ¶13; *see also id.*, ¶34 ("This functionality results in a considerable speed improvement over prior art centralized assembly and delivery of dynamic content."), ¶¶67-69 (edge functionality "significantly reduces the load on the originating site"). Petitioner does not address these teachings, much less explain why a POSA would negate these benefits by moving assembly back to the origin server in Verma.

Petitioner’s reliance on Lewin’s *Background* does not fill this gap. *See* Pet., 57 (citing to EX1007, ¶¶8, 17). Although this section of Lewin discloses an origin system that performs page assembly, it is describing *prior-art systems that did not utilize CDNs*. EX1007, ¶¶8, 17, FIG. 2. Lewin’s system rejects that approach by outsourcing page assembly and delivery to the CDN edge servers to achieve the above-described benefits. EX1007, ¶13. Petitioner again fails to acknowledge this—it does not explain why a POSA would import this prior-art functionality into Verma’s CDN-based system.

To be clear, in Lewin’s system, the application server does perform page assembly upon the very first request for a page, but it does not do so by parsing the HTML for ESI tags. EX1007, ¶¶10, 68. Instead, it simply “performs a database query and assembles the page based on the return values and other common page components, such as navigation menu, logos and advertisement.” EX1007, ¶¶8, 10. It then caches the page at the edge server, such that “all subsequent requests for that page” are assembled and served by the edge servers using the disclosed ESI-based parsing and assembly. EX1007, ¶10. Thus, even where the origin assembles content, it does not perform the claimed tag identification or HTML parsing.

Petitioner argues that the “assembled page” sent to the user “may contain URLs or other CDN-modified resource locators.” Pet., 58 (citing EX1007, ¶50). But this only occurs when the page is assembled by the CDN edge servers. *See*

EX1007, ¶50. The application server does not communicate with the CDN when it first assembles the page, nor does it include CDN-modified resource locators. *See* EX1007, ¶¶8, 10.

Petitioner also argues that Lewin's tags may be implemented on other servers, not just "on an edge server," because Lewin teaches that various "server side scripting language[s]" may be used. Reply, 19; EX1007, ¶49. This is incorrect. Although Lewin's tags may be written using various scripting languages, HTML parsing is consistently performed by the edge server. *See* EX1007, ¶49. Petitioner identifies no other server, let alone an origin server, that performs tag parsing for page assembly.

Ultimately, Petitioner's theory is driven not by the teachings of the references, but by the claims. Petitioner's modification would eliminate the performance and load-reduction benefits of Lewin's edge-based system, while simultaneously transforming Verma's content provider from a routing intermediary into a page assembler. EX1007, ¶¶9, 13, 34, 67-69. Petitioner's lack of explanation for these changes confirms that its combination was formed using improper hindsight. Petitioner therefore fails to establish that a POSA would have combined the references as proposed.

3. Petitioner fails to explain how tag identification triggers retrieval of modified embedded resources.

The POR establishes that Petitioner failed to explain the “connective tissue” for using Lewin’s ESI tags to trigger Verma’s URL-prepend mechanism. POR, 35-39. Petitioner argues that it would have been obvious to do so because “Lewin’s tags initiate fetching content.” Reply, 21-24. Petitioner is wrong.

A POSA would not have used Lewin’s tags to invoke Verma’s URL-prepend mechanism for routing content requests because the two mechanisms serve different functions and operate at different stages of their respective systems. Lewin’s tags are not generic triggers—they specify “dynamic assembly instructions” for building HTML pages. EX1007, ¶49; Reply, 22. Upon identifying a tag, the edge server performs page assembly by fetching HTML chunks (e.g., “us_ad.html,” “canada_ad.html”) and incorporating those chunks into the “page markup.” EX1007, ¶¶13, 43, 50. Thus, Lewin’s tags specify content to be incorporated in a page during edge-based assembly.

Conversely, Verma’s prepended URLs are used to direct client requests to selected cache servers. EX1005, ¶¶25-27. They specify cache servers from which requested content can be retrieved and are generated in response to client requests. EX1005, ¶¶25-26. Accordingly, whereas Lewin’s tags are used to determine *what* content is included in the page, Verma’s prepended URLs determine *where* content is retrieved from.

These differences are reinforced by their positions in the respective communication flows. Lewin's HTML parsing occurs at the edge server, *after the client has already established a connection with the edge server*. EX1007, ¶¶49-50. Verma's prepended URLs, by contrast, are generated at the origin server, specifically to determine which edge server to connect to in the first place. EX1005, ¶¶25-27. A POSA would not have sought to use Lewin's post-routing page-assembly mechanism (tag identification) to trigger performance of Verma's pre-routing network-direction function (server selection).

Petitioner's theory is also internally inconsistent. It relies on the same tags to trigger two distinct functions—Lewin's page assembly and Verma's URL rewrite process. Petitioner's combination implements Lewin's tags for page assembly—i.e., for fetching HTML chunks and incorporating them within the page markup—and it also purports to repurpose *these same tags* to trigger Verma's URL rewrite techniques. Pet., 57; Reply, 22, 24; EX1007, ¶¶13, 50. In other words, Petitioner implements the tags to determine what content is to be included in a page (Lewin) and where that content is to be served (Verma).

The prior art provides no basis for assigning both roles to the same mechanism. Lewin's tags specify HTML fragments or chunks to be implemented within the page markup. EX1007, ¶¶13, 49-50. They do not specify resources whose locations must be determined through CDN routing. That Lewin's "final

HTML page” may include “CDN-modified resource locators” is of no moment.

See Reply, 22, 24; EX1007, ¶50. Lewin’s tags merely trigger retrieval of the HTML chunks, which may happen to include resource locators that have been modified by the CDN. EX1007, ¶50. Identification of these tags does not trigger a modification of these resource locators.

Verma fares no better. In Verma, URL rewriting is triggered when the content provider receives a standard HTTP request specifying a URL. EX1005, ¶¶13, 25-26. This process is thus request-driven and occurs without tag identification. Petitioner does not identify any teaching in Verma suggesting that routing decisions should be invoked through HTML parsing or tag identification.

Petitioner fails to identify a mechanism by which grafting Lewin’s tag-identification concept onto Verma’s request-driven, URL-rewriting system would produce a combined system that actually performs the claimed “upon identification of the tag, using at least one delivery parameter to obtain” sequence. Petitioner has thus not established that the combination of Verma and Lewin renders obvious the “tag” limitations of claim 1.

4. Lewin’s ESI tags do not provide the claimed “embedded resource with a tag.”

It is undisputed that Lewin does not teach the claimed “embedded resource with a tag” under PO’s construction. *See* Reply, 13-14. Petitioner argues only that Verma satisfies this term under its own construction “because the tag is next to,

associated with, and identifies the embedded resource.” Reply, 13. As explained in Section II.C, the Board should adopt PO’s construction. Consequently, the Board should also find that Verma-Lewin does not teach the claimed “embedded resource with a tag.” *See* POR, 28-31.

C. Petitioner fails to establish that the cited art discloses or renders obvious the claimed API.

The POR establishes that Verma fails to teach the claimed API. POR, 39-51. As explained in Section II.B *supra*, the claimed “API” denotes a defined programmatic interface that specifies the inputs a calling system must supply and the output a receiving system will return. EX2015, ¶106; POR, 50-51. Petitioner fails to show that Verma discloses this element under this or any reasonable construction.

The ’322 patent distinguishes API-based communication from other forms of communication not involving an API. EX1001, 3:9-14 (distinguishing “calling an API” from “otherwise querying the CDN”), 7:20-22 (disclosing “other forms of communicating with and passing information to the computing engine”); *see also* Section II.B.2 *supra*. In other words, the ’322 patent discloses communications between the origin and CDN that do not involve an API. Petitioner fails to show that Verma teaches *API-based* communication.

Indeed, Petitioner repeatedly asserts that Verma’s content provider merely “forward[s]” client HTTP requests to the broker. Pet., 2, 15, 17, 19; *see also*

EX1003, ¶¶67, 95, 102, 110; EX1005, claim 9. But, as Dr. Russ explains, forwarding HTTP requests is not the same as invoking an API. EX2015, ¶¶107-108. Petitioner does not contend, nor does Verma disclose, that the client request itself constitutes an API call to the broker. Petitioner fails to show that by *forwarding* such a request, the content provider invokes an API, instead of, for instance, simply forwarding an HTTP request into a system's internal pipeline.

Verma is entirely silent on the structure of the request. It does not specify what parameters the content provider must supply, what interface definition governs the exchange, or what defined output the broker is obligated to return. EX2015, ¶108; EX2016, 36:15-37:12. Verma discloses no defined programmatic interface, nor does Petitioner point to one. EX2015, ¶108.

Instead, Petitioner contends that Verma discloses a “defined interface contract” because the content server “negotiates” with the CDN “through the broker 25,” and because the content provider has a “business relationship” with the CDN. Reply, 26. This misses the mark. Petitioner establishes nothing more than Verma's broker communicates with a content provider pursuant to a business agreement between parties. Petitioner fails to show that this communication is performed through an exposed API—i.e., that it constitutes “calling an API” as opposed to “otherwise querying the CDN.” EX1001, 3:9-14, 7:20-22.

Petitioner further contends that “Verma discloses defined inputs and outputs” because its content provider provides “defined parameters” to the broker and the broker returns a “defined output” in the form of a prepended URL. Reply, 27. But again, this merely shows communication. Verma does not disclose or discuss the structure of the request—including what specific information must be passed to the broker by the content provider. *See* EX1005, ¶¶25-26. That Verma describes information that may be received by the broker in certain requests (e.g., “location and delay tolerance,” EX1005, ¶15) does not evidence use of a defined API for communication.

Accordingly, Petitioner has shown, at most, that Verma involves communication between content provider and broker. That is insufficient. The record evidence fails to establish that Verma’s broker provides the claimed API.

IV. CONCLUSION

For the above reasons, claims 1-15 are patentable over the Grounds asserted in this IPR.

Respectfully submitted,

STERNE, KESSLER, GOLDSTEIN & FOX PLLC

/Jason D. Eisenberg/

Jason Eisenberg, Reg. No. 43,447
Daniel S. Block (Reg. No. 68,395)
Kristina Caggiano Kelly (Reg. No. 58,164)
Steven M. Pappas (Reg. No. 73,904)
Counsel for Patent Owner Sandpiper CDN, LLC

Date: July 13, 2026

1101 K Street, NW, 10th Floor
Washington, DC 20005
(202) 371-2600

CERTIFICATE OF WORD COUNT (37 C.F.R. § 42.24(d))

1. This Patent Owner Sur-Reply to Petitioner's Reply complies with the type-volume limitation of 5,600 words, comprising 5,600 words, excluding the parts exempted by 37 C.F.R. § 42.24(a)(1).

2. This Patent Owner Sur-Reply to Petitioner's Reply complies with the general format requirements of 37 C.F.R. § 42.6(a) and has been prepared using Microsoft® Word in 14-point Times New Roman font.

Respectfully submitted,

STERNE, KESSLER, GOLDSTEIN & FOX PLLC

/Jason D. Eisenberg/

Jason D. Eisenberg
Registration No. 43,447
Counsel for Patent Owner Sandpiper CDN, LLC

Date: July 13, 2026

1101 K Street, NW, 10th Floor
Washington, DC 20005
(202) 371-2600

CERTIFICATE OF SERVICE (37 C.F.R. § 42.6(e))

I certify that the above-captioned **PATENT OWNER'S SUR-REPLY TO PETITIONER'S REPLY** was served in its entirety on July 13, 2026, upon the following parties via electronic mail:

Erika H. Arner (Lead Counsel) erika.arner@finnegan.com
Kara A. Specht (Back-Up Counsel) kara.specht@finnegan.com
Sydney R. Kestle (Back-Up Counsel) sydney.kestle@finnegan.com
Cara E. Regan (Back-Up Counsel) cara.regan@finnegan.com
Milan F. Feliciello (Back-Up Counsel) milan.feliciello@finnegan.com
Daniel C. Tucker (Back-Up Counsel) daniel.tucker@finnegan.com
Google-Sandpiper-IPRs@finnegan.com
FINNEGAN, HENDERSON, FARABOW, GARRETT & DUNNER, LLP

Respectfully submitted,

STERNE, KESSLER, GOLDSTEIN & FOX PLLC

/Jason D. Eisenberg/

Jason D. Eisenberg
Registration No. 43,447
Counsel for Patent Owner Sandpiper CDN, LLC

Date: July 13, 2026

1101 K Street, NW, 10th Floor
Washington, DC 20005
(202) 371-2600