

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 RESPONSE
UNDER 37 C.F.R. § 42.120**

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	1
II.	BACKGROUND	3
III.	THE '322 PATENT	6
	A. The '322 patent provides novel techniques for improving the performance of content delivery networks.....	6
	B. Prosecution History	8
	C. Level of Ordinary Skill in the Art	11
	D. Claim Construction.....	11
	1. “direct link”	11
IV.	OVERVIEW OF THE PRIMARILY RELIED ON REFERENCES.....	15
	A. Verma	15
	B. Raciborski.....	17
	C. Lewin.....	19
V.	THE PETITION FAILS TO ESTABLISH ANTICIPATION OR OBVIOUSNESS OF ANY CHALLENGED CLAIM.	19
	A. Verma, alone or in combination with Raciborski or Lewin, does not disclose or teach a “direct link” to a CDN node as required by all challenged claims.	21
	1. Petitioner’s characterization of Verma’s “prepended URL” as a “direct link” is incorrect; Verma’s “akamaized” URLs are DNS-dependent hostname URLs, not “direct links.”	22
	2. Raciborski requires DNS resolution and thus does not cure Verma’s “direct link” deficiency (Grounds 2, 4, 5).....	25
	3. Lewin also requires DNS resolution and thus does not cure Verma’s “direct link” deficiency (Grounds 3, 4, 5).....	26
	B. The Petition fails to establish that Verma, alone or in combination with Lewin, renders obvious the “tag” limitations of independent claim 1 (Grounds 3-5).	28
	1. Lewin’s ESI tags do not provide the claimed “embedded resource with a tag.”	28

2.	Petitioner has not established a motivation to combine Verma and Lewin.....	31
3.	Petitioner fails to explain how tag identification in the combined system triggers the origin system’s use of delivery parameters to obtain a modified embedded resource.	35
C.	The Petition fails to establish that the cited art discloses or renders obvious the claimed “application programming interface (API)” (Grounds 1A/1B, 2, 4; Claims 11 and 15).	39
1.	Claims 2, 9, and 11-15 require an exposed “API” to receive requests, regardless of whether the API includes additional functionality.	40
2.	Petitioner has not shown that Verma’s broker provides the claimed “API.”	46
3.	Dr. Mowry’s testimony and the Board’s preliminary reliance on Dr. Russ’s general API definition do not fill the gaps in Verma.	49
VI.	CONCLUSION.....	52

PATENT OWNER’S UPDATED EXHIBIT LIST

Exhibit No.	Description
2001	“Cloud CDN: content delivery network,” <i>Google Cloud</i> , accessed at https://cloud.google.com/cdn (last accessed July 2, 2025)
2002	Interim Processes for PTAB Workload Management Memorandum (March 26, 2025)
2003	National Judicial Caseload Profile – United States District Courts (March 2025)
2004	<i>Intentionally Left Blank</i>
2005	<i>Sandpiper CDN, LLC v. Google LLC</i> , No. 2:24-cv-03951, Invalidity Contentions (April 22, 2025)
2006	<i>Sandpiper CDN, LLC v. Google LLC</i> , No. 2:24-cv-03951, ECF No. 82 (C.D. Cal. June 24, 2025)
2007	<i>Sandpiper</i> , No. 2:24-cv-03951, ECF No. 28 (C.D. Cal. Sept. 16, 2024)
2008	FAQs for Interim Processes for PTAB Workload Management, accessed at https://www.uspto.gov/patents/ptab/faqs/interim-processes-workload-management
2009	Interim Director Discretionary Process, accessed at https://www.uspto.gov/patents/ptab/interim-director-discretionary-process
2010	Declaration of Samuel H. Russ, Ph.D.
2011	<i>Curriculum Vitae</i> of Samuel H. Russ, Ph.D.
2012	Goodwin, M., “What is an API (application programming interface)?” accessed at https://www.ibm.com/think/topics/api (last accessed August 5, 2025)

Exhibit No.	Description
2013	“CDN Stands For Content Delivery Network: History and Evolution,” <i>BlazingCDN</i> , accessed at https://blog.blazingcdn.com/en-us/cdn-stands-for-content-delivery-network-history-and-evolution (last accessed August 1, 2025)
2014	“What is a CDN (Content Delivery Network)?” <i>Amazon Web Services</i> , accessed at https://aws.amazon.com/what-is/cdn/ (last accessed August 1, 2025)
2015	Declaration of Samuel H. Russ, Ph.D. in support of Patent Owner’s Response
2016	Deposition Transcript of Todd C. Mowry, February 25, 2026
2017	“ARL syntax,” Akamai Techdocs, available at https://techdocs.akamai.com/edge-diagnostics/docs/arl-syntax (last accessed February 3, 2026)
2018	Cohen, E. & Kaplan, H., “The Age Penalty and its Effect on Cache Performance,” <i>3rd USENIX Symposium on Internet Technologies and Systems</i> (2001)
2019	“What Is an API (Application Programming Interface)?” Oracle, available at https://www.oracle.com/cloud/cloud-native/api-management/what-is-api/ (last accessed March 3, 2026)
2020	“What’s the difference between APIs and a simple http request?,” Medium, available at https://medium.com/@paul.k.pallaghy/whats-the-difference-between-apis-and-a-simple-http-request-c31692355dd1
2021	“Connect – Most Innovative New Products Award Winners Announced,” Connect NewsLetter, December 16, 1999, available at https://library.ucsd.edu/dc/object/bb60436752/_1.pdf (last accessed March 1, 2026)

Exhibit No.	Description
2022	“Sandpiper Named ‘Best of the Best’ at Networld + Interop 99 Las Vegas,” PRNewswire, May 12, 1999, available at https://ubmchannelnews.mediaroom.com/index.php?s=29573&item=85841 (last accessed March 1, 2026)
2023	Entrepreneur of the Year, June 19, 2000
2024	“Sandpiper Networks’ Footprint Takes Aim at Internet Traffic,” Wall Street Journal Interactive Edition, June 17, 1999, available at https://url.us.m.mimecastprotect.com/s/wTVWCxklA1iDQQWjSYi2Syt1OK?domain=wsj.com (last accessed March 3, 2026)
2025	“Web-Page Distribution System Could Unclog Internet Traffic Jams,” The New York Times, April 26, 1999, available at https://archive.nytimes.com/www.nytimes.com/library/tech/99/04/biztech/articles/26sand.html (last accessed March 3, 2026)
2026	“Sandpiper Puts Its Footprint in Marketplace,” Los Angeles Times, September 28, 1998, available at https://www.latimes.com/archives/la-xpm-1998-sep-28-fi-27201-story.html

I. INTRODUCTION

U.S. Patent No. 10,057,322 (“the ’322 patent”) provides novel techniques for distributing content to end users via content delivery networks (CDNs). These techniques involve an alternative to using conventional domain name system (DNS) servers for edge selection by instead calling an API, or otherwise querying the CDN, when constructing a uniform resource locator (URL) for the resource to be delivered from the CDN, improving the performance of CDNs. In the end, all Grounds rely on a flawed analogy between the U.S. Publication No. 2009/0132640 (“Verma”) content delivery system and the ’322 patent because Verma discloses a disparate technical architecture and operation.

Petitioner presents five Grounds challenging the claims of the ’322 patent:

Ground	Art	Basis	Claims (Independent Bolded)
1A	Verma (EX1005)	§ 102	15
1B	Verma	§ 103	11, 12, 14, 15
2	Verma and Raciborski (EX1006)	§ 103	11, 12, 14, 15
3	Verma and Lewin (EX1007)	§ 103	1, 8, 10, 13
4	Verma, Raciborski, Lewin	§ 103	2, 4, 5, 7, 9, 13
5	Vermin, Raciborski, Lewin, Draï (EX1008)	§ 103	3, 6

Pet., 10. Petitioner fails to demonstrate that any of these claims are anticipated or rendered obvious in view of the cited art.

First, for all grounds, Petitioner fails to demonstrate that Verma (alone or in combination with Raciborski or Lewin) teaches “a direct link to a node within the

content delivery network,” as recited in every independent claim. Petitioner points to Verma’s “prepending” URLs to satisfy the claimed “direct link.” Pet., 23-24.

But, unlike the claimed “direct link,” each of those URLs undisputedly includes a hostname that requires conventional DNS resolution. That is fundamentally at odds with the ’322 patent’s stated goal of providing “an alternative to using conventional domain name system (DNS) servers for edge selection.” EX1001, 3:9-14.

Second, for Grounds 3-5 (independent claim 1), Petitioner fails to establish that Verma, alone or in combination with Lewin, renders obvious the claimed “tag” limitations. Petitioner acknowledges that Verma does not disclose the claimed “tag” limitations, instead relying on Lewin. Pet., 45-48. Lewin, however, does not disclose an “embedded resource with a tag,” as required by independent claim 1. In fact, Lewin’s purported “embedded resource” (a URL) does not include a tag at all. Petitioner also fails to establish a motivation to combine Verma and Lewin, or that a person of ordinary skill in the art (“POSITA”) would have had a reasonable expectation of success in achieving the combination.

Finally, for Grounds 1A/1B, 2, and 4 (claims 2, 9, and 11-15), Petitioner fails to establish that Verma renders obvious the claimed “application programming interface (API).” Petitioner relies solely on Verma’s content broker for the recited “application programming interface (API)” across all Grounds, but

Verma's content broker is described as nothing more than an intermediary between two other devices. Verma is devoid of any discussion regarding its content broker exposing an API, and neither Petitioner nor its expert fill the gaps of Verma's silent disclosure.

The Petition, therefore, fails to satisfy its burden of proof to demonstrate that any independent claim is anticipated or obvious over Verma, alone or in combination with Raciborski or Lewin. The Board should find all challenged claims patentable.

II. BACKGROUND

The '322 patent discloses pioneering technology within content delivery networks ("CDNs"). CDNs provide critical services that enable content providers to efficiently deliver content to end users over the Internet. EX2015, ¶26. In the early 1990s, a California start-up called Sandpiper Software Consulting, LLC, a predecessor of Patent Owner, improved the operation of CDNs through a series of innovations.

A typical server in the 1990s could only handle a limited number of simultaneous connections from the growing number of World Wide Web users before becoming overloaded. EX2015, ¶27. Moreover, end users experienced sluggish load times and high latency due to problems such as congested network segments and geographic separation between the users and physical servers. *Id.*

Sandpiper Software Consulting, LLC was among the first to develop services that allowed content providers to distribute their content over the Internet using a decentralized architecture. The solution overcame the common congestion and performance issues that plagued Internet transmission by deploying CDN servers around the world. EX2015, ¶¶27-29. The solution replicated content from customers' origin servers to appropriate CDN servers and transparently rendezvoused end users requesting that content to the best CDN server. *Id.* Others in the industry quickly imitated this service and architecture. *Id.*

Using these solutions, end users would connect to an optimal edge server—i.e., a closest server having available capacity to serve their request. EX2015, ¶29. Distributing content across a network of servers alleviated data congestion issues and reduced latency. *Id.* The solution further involved propagating data from origin servers to edge servers (known as “caching”) based on network demand and seamlessly routing users to the optimal edge server with the correct content. Using this type of caching to perform the replication process within the CDN framework contrasted and improved conventional content retrieval architectures. *Id.*

In 1996, Sandpiper Networks Inc. (“Sandpiper Networks”) was founded in Thousand Oaks, California to further develop and commercialize their novel concept for a content delivery network. There, the inventors designed and built a CDN ultimately referred to as “Footprint.” By at least May 24, 1996, the Sandpiper

Networks team had conceived techniques for delivering static and dynamic content, including streaming audio and video, using Sandpiper's CDN.

By May 1998, Sandpiper Networks delivered cached content to end users using its CDN. Sandpiper Networks' first paying customer was the L.A. Times, which paid Sandpiper Networks to host the report of Independent Counsel Ken Starr on his investigation of President Bill Clinton ("the Starr Report") beginning on September 11, 1998. Sandpiper Networks' CDN was capable of caching, and was used to cache and deliver Internet resources including, among other things, pictures, text files, dynamic resources, and streaming multimedia resources.

Sandpiper Networks labored not only to build and implement its CDN, but also to protect its groundbreaking innovations through the patent system. Recognizing that its inventions could revolutionize content delivery worldwide, Sandpiper Networks—and its successor companies, including Patent Owner—filed numerous patent applications directed to its foundational CDN technology, including the '322 patent.

The industry-wide acclaim for Sandpiper Networks further confirms the non-obvious nature of its innovations. Sandpiper's pioneering CDN technology was honored with the "Best of the Best" award at Networld+ Interop 99 Las Vegas, and the 1999 Most Innovative New Product Award from the University of California, San Diego (UCSD) CONNECT program. EX2021; EX2022. This

technical leadership was mirrored by executive recognition, including the Ernst & Young 2000 Entrepreneur of the Year Award bestowed upon Andrew Swart and David Farber, founders of Sandpiper Networks. EX2023. Beyond formal awards, Sandpiper's disruption of the CDN space was extensively documented by premier national publications, including The Wall Street Journal, The New York Times, and The Los Angeles Times, all of which highlighted the company's novel approach to content delivery. EX2024; EX2025; EX2026.

In the early-to-mid 2000s, because of Sandpiper Networks' innovations, demand for CDNs exploded. EX2015, ¶30. Content providers like Google and YouTube use CDN technology to meet their enormous data streaming needs, as well as for video search results. EX2001, 1-3; EX2015, ¶30. As a result, techniques for controlling how users access content on a CDN became increasingly important. EX2015, ¶30.

III. THE '322 PATENT

A. The '322 patent provides novel techniques for improving the performance of content delivery networks.

The '322 patent is entitled "Network Address Resolution," and it claims priority to U.S. Provisional Application No. 62/098,930 ("the '930 application"), filed on December 31, 2014.

The '322 patent is directed to novel techniques for improving the performance of content delivery networks (CDNs), in some cases using an

application programming interface (API) to facilitate embedding of “link resolution information within an HTML document or other content returned to a requesting client.” EX1001, 1:13-18. Aspects of the ’322 patent “involve an alternative to using conventional domain name system (DNS) servers for edge selection by instead calling an API or otherwise querying the CDN when constructing a uniform resource locator (URL) for the resource to be delivered from the CDN.” EX1001, 3:9-14. The solutions described in the ’322 patent were unconventional at the time of the invention of the ’322 patent and continue to offer numerous technical benefits. EX2015, ¶31.

For example, as users were able to access more content (e.g., television, movies, news, software updates) on networks, network providers sought alternatives to routing requests through DNS to select an optimal edge server. EX1001, 1:39-50. The inventors of the ’322 patent devised techniques for “bypassing DNS,” thereby reducing the number of components involved in servicing requests while improving response speeds when delivering content. EX1001, 2:45-49, 5:29-32, 6:60-7:1; EX2015, ¶¶32-33. And these techniques are recited in the claims of the ’322 patent.

For instance, Petitioner-chosen representative independent claim 15 recites:

A method comprising:

providing, at a content delivery network, an application programming interface (API);

at the API, receiving, from an origin system that is separate from the content delivery network, an embedded resource and at least one parameter associated with the embedded resource;

generating a modified embedded resource providing a direct link to a node within the content delivery network for obtaining content associated with the embedded resource;

delivering, by the content delivery network, the modified embedded resource to the origin system;

receiving, at the node, a request from a client device to obtain the content; and

providing the content to the client device.

EX1001, 12:5-20.

The solutions described and claimed in the '322 patent improve the functioning of a CDN, including, for example, offloading computational resource consumption away from a client-side DNS infrastructure, and instead having content retrieved from an optimal node in the network. EX2015, ¶¶34-35.

B. Prosecution History

The '322 patent was filed as U.S. Application No. 14/985,968 (“the '968 application”) on December 31, 2015. The '322 patent claims priority to U.S. Provisional Application No. 62/098,930, filed December 31, 2014. The '322 patent received a thorough prosecution on the merits, including overcoming both §§ 101 and 102 rejections. EX1002, 15-18, 60-74, 95-108.

After receiving a non-final rejection, *see* EX1002, 95-108, Patent Owner amended the independent claims to clarify the architecture of the claimed systems and methods. EX1002, 56-59. For example, Patent Owner amended independent claim 12 (which issued as independent claim 11) as follows:

12. (Currently Amended) A content delivery ~~device~~ network comprising:
at least one processor in communication with a non-transitory computer readable medium including computer readable instructions for:
- ~~generating~~ providing an application programming interface (API);
 - at the API, receiving, from an origin system that is separate from the content delivery network, an embedded resource and at least one parameter associated with the embedded resource;
 - generating a modified embedded resource providing ~~information as to a location within a~~ a direct link to a node within the content delivery network for obtaining content associated with the embedded resource;
 - delivering, by the content delivery network, the modified embedded resource to the origin system;
 - receiving, at the node, a request from a client device to obtain the content; and
 - providing the content to the client device.

EX1002, 57-58.

Patent Owner also added independent claim 22 (which issued as independent claim 15) with similar limitations:

22. (New) A method comprising:

providing, at a content delivery network, an application programming interface (API);

at the API, receiving, from an origin system that is separate from the content delivery network, an embedded resource and at least one parameter associated with the embedded resource;

generating a modified embedded resource providing a direct link to a node within the content delivery network for obtaining content associated with the embedded resource;

delivering, by the content delivery network, the modified embedded resource to the origin system;

receiving, at the node, a request from a client device to obtain the content; and

providing the content to the client device.

EX1002, 59.

Patent Owner argued that the claimed architecture differed from the prior art (U.S. Publication No. 2014/0289319 to Richardson) because “Richardson require[d] the client to either (a) communicate with the DNS server of the CDN to determine the node that will be used to deliver the content; or (b) execute translation code to ask the CDN for translation information, which translation information may include the address of the node that will be used to deliver the content.” EX1002, 72-73. Richardson thus did not disclose “generating a modified embedded resource providing a direct link to a node within the content delivery network for obtaining content associated with the embedded resource.” *Id.*;

EX2015, ¶¶36-38. Upon consideration of this and other arguments, the Examiner allowed the claims. EX1002, 15-18.

As discussed below, the prior art presented in the Petition also fails to disclose or render obvious the limitations of the independent claims found allowable by the Examiner.

C. Level of Ordinary Skill in the Art

A POSITA at the time of the '322 patent would have had at least a bachelor's degree in computer science, electrical engineering, or other suitable field, and at least two years of work or research experience related to the field of content delivery networks. EX2015, ¶25. Further professional experience could substitute for education, and vice versa. *Id.*

D. Claim Construction

While Patent Owner believes that all claim terms should be accorded their plain and ordinary meaning as understood by a POSITA at the time of the invention in light of the specification and the prosecution history, Petitioner's interpretation of the term "direct link" is at odds with the '322 patent's specification and the prosecution history. That term is discussed further below.

1. "direct link"

Each independent claim of the '322 patent recites a "modified embedded resource" that includes or provides "a direct link to a node within the content delivery network." EX1001, 10:5-18 (Claim 1) ("wherein the modified embedded

resource includes a *direct link*¹ to a node within the content delivery network that can deliver content associated with the embedded resource”), 10:59-11:9 (Claim 11) (“generating a modified embedded resource providing a *direct link* to a node within the content delivery network for obtaining content associated with the embedded resource”), 12:5-20 (Claim 15) (“generating a modified embedded resource providing a *direct link* to a node within the content delivery network for obtaining content associated with the embedded resource”). In the context of the ’322 patent specification and prosecution history, the term “direct link” refers to a network address that enables a client device to bypass conventional DNS resolution of a hostname when contacting a CDN node. This understanding is well supported by the intrinsic record. EX2015, ¶¶41-46.

In particular, the ’322 patent specification confirms that a “direct link” is one that allows the client device to bypass conventional DNS resolution. *Phillips v. AWH Corp.*, 415 F.3d 1303, 1315–17 (Fed. Cir. 2005) (en banc) (the specification is “the single best guide to the meaning of a disputed term”); *Slimfold Mfg. Co. v. Kinkead Indus., Inc.*, 810 F.2d 1113, 1116 (Fed. Cir. 1987) (“Claims are not interpreted in a vacuum, but are part of and are read in light of the specification.”); EX2015, ¶42. For example, Figure 1 of the ’322 patent provides “a network

¹ Emphasis added unless otherwise indicated.

diagram and message flow related to generating an embedded resource for inclusion in a network asset, such as an HTML document, the embedded resource *providing a connection to a node of a CDN bypassing DNS.*” EX1001, 2:47-49.

The ’322 patent explains that “[t]he location of the client may be used in selecting an optimal node to serve a request for the embedded resource,” and “by *embedding an IP address* to serve the content directly, *the client device may bypass DNS and instead connect directly to the source of the embedded resource.*” EX1001, 5:27-32. In other words, the specification expressly equates “directly” connecting with bypassing DNS through an embedded IP address. *Id.* Based on this disclosure, a POSITA would have understood the “direct link” to be one that provides the client with an address enabling it to initiate a connection to the CDN without first performing a conventional DNS lookup. EX2015, ¶42.

As Dr. Russ explains, DNS resolution is the standard mechanism by which a hostname is resolved to the IP address required to establish a TCP connection, and bypassing that process is a key technical improvement described by the specification. EX1001, 5:29-32; EX2015, ¶43. Indeed, the ’322 patent specification premises its disclosure on this improvement, describing that “[a]spects of the present disclosure involve *an alternative to using conventional domain name system (DNS) servers for edge selection.*” EX1001, 3:9-14. Notably,

the Petition also relies on this very sentence to characterize the '322 patent, acknowledging the '322 patent's focus on DNS-bypass technology. Pet., 8.

The remaining description and embodiments provided in the specification confirm that a “direct link” is one that allows the client device to bypass conventional DNS resolution. EX2015, ¶44. Figure 3, for example, includes operation 330 that returns a “network address of [an] identified node” to the origin server for inclusion as the embedded resource. EX1001, FIG. 3 (Operation 330). The associated description provides that “[t]he API returns an absolute URL from which the content may be retrieved,” which may include an “IP address[]” so that “[w]hen the user selects the link, the browser will *directly contact the node* for the content.” EX1001, 6:50-62; EX2015, ¶44. And in the case that the “absolute URL” includes “a hostname,” the patent explains that “the embedded resource may include specific reference to the CDN such that name servers 34 of the CDN are *contacted directly, bypassing any other DNS processing.*” EX1001, 6:50-7:1; EX2015, ¶44. In other words, regardless of the form of the “absolute URL” included in the embedded resource, the '322 patent describes that the link allows for bypassing of conventional DNS processing. The CDN is contacted directly in all instances, in line with the '322 patent's stated goal of providing “an alternative to using conventional domain name system (DNS) servers for edge selection.” EX1001, 3:9-14; EX2015, ¶44.

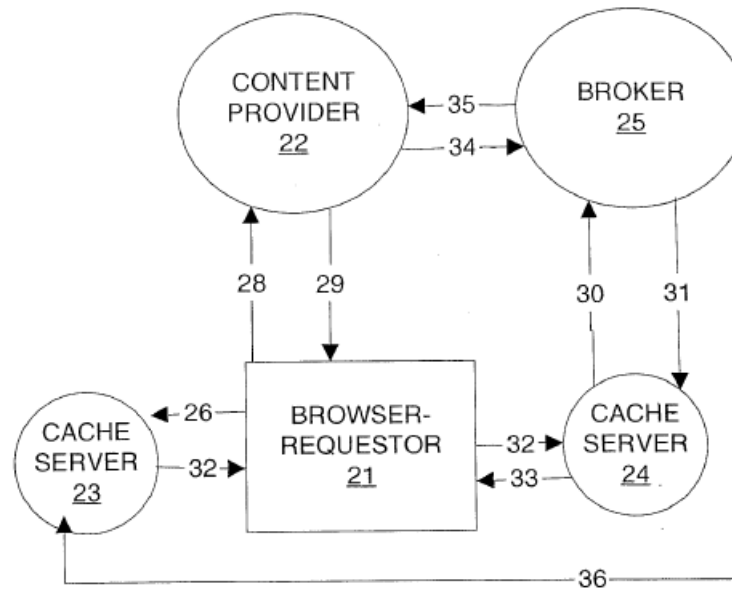
Petitioner’s implicit interpretation of “direct link” as encompassing any complete hostname-based URL—including one requiring conventional DNS resolution—ignores the DNS-bypass requirement of a “direct link” described in the ’322 patent specification. *See* Pet., 23 (asserting that a “direct link” can be met merely by a link that provides a “path bound to a single cache server node”). That a URL may map to one particular cache server is not enough to satisfy the claimed “direct link”—the ’322 patent specification does not support such a broad interpretation of “direct link” that would allow a URL to be resolved through the conventional DNS hierarchy. EX2015, ¶45.

Accordingly, in view of the ’322 specification, 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.

IV. OVERVIEW OF THE PRIMARILY RELIED ON REFERENCES

A. Verma

Verma discloses a “computer method and related apparatus for delivering content files to a client computer.” EX1005, Abstract, ¶¶10-11. The method includes modifying a resource link—based on client location and content availability—to direct requests to a cache server node in a CDN. EX2015, ¶47. Figure 1 shows an exemplary “content delivery network.”



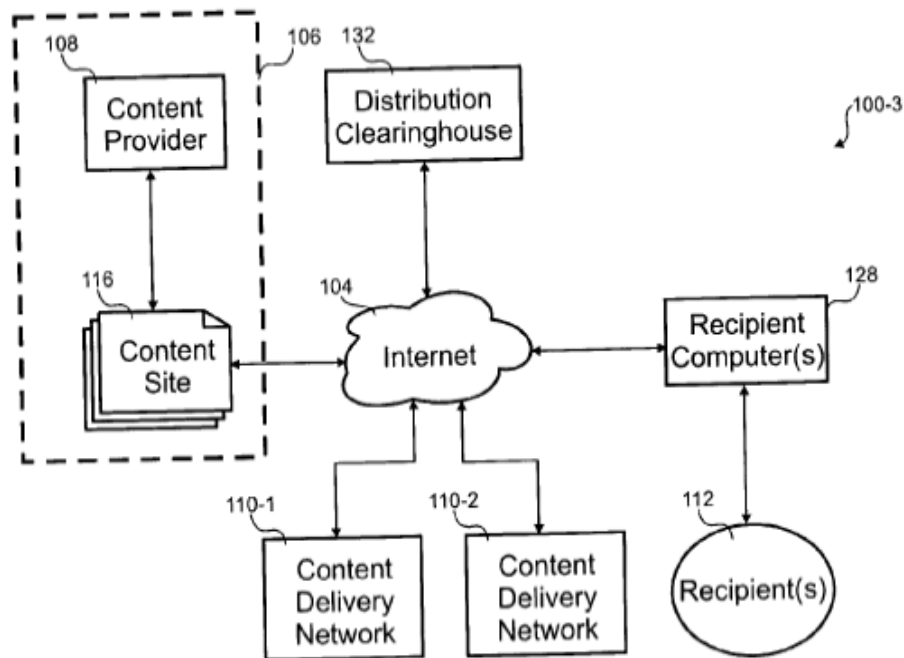
EX1005, FIG. 1.

In Verma, the content delivery network includes a client that runs a “browser 21,” a “content provider 22,” a “content delivery network broker 25,” and “a set of cache servers” “23”/“24.” EX1005, FIG. 1, ¶¶22, 25; EX2015, ¶48. The client “browser 21 operates to request a content file ... by sending a URL 28 to a content provider 22.” EX1005, ¶25, FIG. 1. “The request from the client ... can comprise delay tolerance information” EX1005, ¶18. This request is then “forward[ed]” from content provider 22 to the content delivery network broker 25 that “manages at least one, and preferably, a set of cache servers.” EX1005, claim 9, ¶25. Also, content provider 22 can “dynamically request the broker 25 for a content file delivery to a location close to the client.” EX1005, ¶26. The broker sends availability inquiries to the cache servers to “find out whether they currently store

the requested” file. EX1005, ¶25. The broker 25 can select the appropriate cache server to serve the content and return a “re-written URL to the content server [that] received the [client] request.” EX1005, ¶¶12, 26-27, 37. When the client receives the prepended (re-written) URL, “it requests the content file from the appropriate cache server when the content is available.” EX1005, ¶37; EX2015, ¶48.

B. Raciborski

Raciborski discloses a “content delivery network (CDN) for delivering content objects from a content provider to a recipient.” EX1006, Abstract; EX2015, ¶49. Figure 1C shows an exemplary network.



EX1006, FIG. 1C.

Within this network, Raciborski discloses a “content provider” that “sends various information to the one or more CDNs” in the network to facilitate content delivery. EX1006, ¶¶26, 55-56. The information can include the “IP address of the recipient computer,” the “full object location (i.e., server, path and name),” “if delivery can be delayed,” and a “protocol used to deliver the content object,” among other things. *Id.* This information may be included in a “bid message,” like the one illustrated in Table 1. EX1006, ¶¶55-56, Table 1.

TABLE I	
<u>Real Time Bid Message Example</u>	
Field	Example
Bid Identifier	34sd89732k
Recipient IP Address	69.93.18.222
Domain of Content Object	LLNW.com
Path of Content Object	/press/2005releases/
File Name of Content Object	routemark.mp4
Format	MPEG4
Bitrate	200 Kbps
Latency Options	3
QoS Level(s)	3
Packet Loss Options	2
Buffer Duration	24 hours
Max. Volume	1000

EX1006, Table 1.

Such “bid message” parameters, as illustrated in Table 1, provide the CDN with information it can use to select an optimal delivery path. EX1006, Table 1; EX2015, ¶50.

C. Lewin

Lewin is generally directed to a “dynamic content assembly mechanism” for constructing an “HTML page.” EX1007, ¶¶8, 12, 28, 49. As Lewin explains, “[t]raditionally,” “directives [] are placed in HTML pages” “that are evaluated on a server before the page is served.” EX1007, ¶28. These “directives” or “dynamic assembly instructions” may be “URLs for HTML chunks to be incorporated in[to] the final HTML page.” EX1007, ¶¶49, 58 (“dynamic content assembly page directives”). These “URLs” are “specif[ied]” by “tags” that can be identified when a server “parses the HTML page.” *Id.* Lewin’s “tags,” however, are not part of Lewin’s URLs like they are in the ’322 patent. EX1007, ¶¶28, 30; EX1001, 4:32-38; EX2015, ¶51.

V. THE PETITION 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.

First, the Petition fails to account for the meaning of the term “direct link” as it is used in the ’322 patent specification and claims (all Grounds). As explained below, the intrinsic record compels the conclusion that a “direct link to a node within the content delivery network” is a network address that enables a client device to connect to a CDN node without requiring conventional DNS resolution

of a hostname. No asserted reference, alone or in combination, discloses or suggests such a link. Because this limitation appears in every independent claim from which the fifteen challenged claims depend, this deficiency is dispositive of every instituted ground.

Second, claims 11 and 15 (and their dependents) recite that the content delivery network provides an “application programming interface (API)” through which the origin system transmits an embedded resource and associated parameters to the CDN (Grounds 1A, 1B, 2, and 4). The Petition maps this limitation to Verma’s internal content broker. But, as explained below, the ’322 patent specification passage on which Petitioner and the Board relied on to find that the broker satisfies the API limitation describes an unclaimed alternative embodiment—not the API architecture that the claims recite. Verma’s broker does not provide the claimed API.

Third, the Petition fails to establish that Verma, alone or in combination with Lewin, renders obvious the claimed “tag” and “upon identification of the tag ... obtain” limitations (Grounds 3-5) for three independent reasons. First, Lewin’s ESI tags do not perform the claimed function. Second, Petitioner has not established that a POSITA would have been motivated to combine Verma with Lewin. Third, even if combined, the Petition and Dr. Mowry fail to explain how tag identification in the combined system triggers the origin system’s use of delivery parameters to

obtain a modified embedded resource, which independently defeats both the teaching of element [1b] in the references and any showing of reasonable expectation of success of the combination. Each of these failures independently defeats Grounds 3-5.

A. Verma, alone or in combination with Raciborski or Lewin, does not disclose or teach a “direct link” to a CDN node as required by all challenged claims.

Claim 1 recites “wherein the modified embedded resource includes a direct link to a node within the content delivery network that can deliver content associated with the embedded resource.” EX1001, 10:5-18 (Claim 1). Claim 11 recites “generating a modified embedded resource providing a direct link to a node within the content delivery network for obtaining content associated with the embedded resource.” EX1001, 10:59-11:9 (Claim 11). Claim 15 recites “generating a modified embedded resource providing a direct link to a node within the content delivery network for obtaining content associated with the embedded resource.” EX1001, 12:5-20 (Claim 15). Every challenged claim depends from, or is, one of these three independent claims. If the asserted references do not teach a “direct link” as that term is properly construed, all five Grounds fail.

As discussed in Section III.D, a “direct link” in the context of the ’322 patent refers to a network address that enables a client device to bypass

conventional DNS resolution of a hostname when contacting a CDN node. No asserted reference—alone or in combination—discloses such an address.

1. Petitioner’s characterization of Verma’s “prepended URL” as a “direct link” is incorrect; Verma’s “akamaized” URLs are DNS-dependent hostname URLs, not “direct links.”

The Petition contends that Verma’s “‘prepended URL’ (‘rewritten URL’) provides a direct link to a CDN node for obtaining content associated with the embedded resource, as it ‘comprises a link to a content server or a cache server’ allowing a client to directly contact the node to access requested content.” Pet., 23-24; *see also* EX1003, ¶¶119-121 (Dr. Mowry testifying that Verma’s prepended URL “provides a complete path (direct link) to a ‘cache server’ (node)”). Petitioner’s argument equates “direct link” with any URL that, when followed, ultimately reaches a particular cache server. This construction reads “direct link” out of the claims because it would be satisfied by *any* URL pointing to a server—including an ordinary hostname-based URL that requires full public DNS resolution—thereby rendering the term meaningless. EX2015, ¶55.

Verma’s primary mechanism is “prepending” (also called “akamaizing”) original content URLs. EX1005, ¶¶13, 26-27; EX2015, ¶56. Verma explains that its technique is “to replace the original URL (e.g. http://contentURL) with an Akamai-ed URL (e.g. http://local_cache_server/contentURL) that redirects to a local cache server.” EX1005, ¶13; *see also id.*, ¶26 (“the content file URLs are

prepended (Akamai-ed)"). The resulting URL—for example, “http://location1/contentURL” or “http://location2/contentURL,” EX1005, ¶29—substitutes a CDN *hostname* for the original host. See EX1003, ¶¶49, 149, 151 (Petitioner’s expert Dr. Mowry testifying that each “prepended URL[]” in Verma includes “a hostname for a cache server, ‘location1,’ ‘location2,’ or ‘local_cache_server’”). It does not contain an IP address or any other form of DNS-bypassing network address. EX2015, ¶56.

This technical distinction is critical in the context of the ’322 patent. EX2015, ¶57. A generic hostname-based URL (e.g., “http://edge.cdn.example.net/content.mp4”) requires the client’s browser to perform a DNS lookup to resolve the hostname to an IP address before any TCP connection can be initiated. An IP-literal URL (e.g., “http://112.0.2.1/content.mp4”) allows the client to open a TCP connection directly to the specified IP address, bypassing DNS entirely. EX2015, ¶57. A POSITA would have readily understood this distinction as fundamental to network addressing, because the public domain name system exists precisely to translate human-readable hostnames into the numerical IP addresses that network protocols require for routing, and whether or not a client must invoke that translation is a basic architectural characteristic of any network addressing scheme. *Id.*

Verma's akamaiized URLs are unambiguously hostname-based. EX2015,

¶58. Every example URL in Verma uses a stand-in hostname

("local_cache_server," "location1," "location2") as the host portion of the URL.

EX1005, ¶¶13, 29, 31. As both experts agree, Akamai's URL rewriting technique used by Verma uniformly produces hostname-based URLs using hostnames that ultimately direct users to Akamai edge server clusters having "Akamai domain[s]." EX2017, 1; EX2015, ¶58 (citing EX2017); EX2016, 44:1-47:17 (confirming "Akamai-style URL rewriting" as used in Verma provides a "URL with [a] host name"). Indeed, Dr. Mowry's own evidence explains that Akamai-style URL rewriting produces a new URL with an altered *hostname* associated with the CDN, for example

"http://ghost467.ghosting.akamai.com/www.provider.com/TECH/images/space.story.gif>space. story.gif." EX1003, ¶149 (citing EX1009, 8:2-12); EX2015, ¶60.

And this rewritten URL requires use of DNS hierarchical processing "to resolve the altered hostname." EX1009, 9:8-47; EX2016, 55:11-57:8 (explaining that resolution of the hostname "would use the standard techniques that we use with network accesses," and "[t]he translation of a host name to an IP address is -- on the Internet, is typically done through DNS."); EX2015, ¶¶58-60 (explaining Akamai's "URL substitution architecture").

Nowhere does Verma disclose an IP address or any other form of DNS-bypassing address, nor does Petitioner suggest it does. EX2015, ¶61. This absence is not incidental; it reflects the fundamental architecture of Verma’s system. Unlike the “direct link” of the ’322 patent, Verma’s entire routing model depends on DNS resolution of CDN hostnames to direct clients to the appropriate edge server. EX1005, ¶¶12-13, 26-27; EX2015, ¶61. And replacing Verma’s hostname-based URLs with static IP-literal addresses would undermine Verma’s load-balancing and content-scheduling architecture as the CDN would no longer be able to leverage DNS-based traffic steering to dynamically route client requests to optimal edge servers. EX1005, ¶¶12-13, 26-27; EX2015, ¶61.

Accordingly, the Petition fails to establish that Verma anticipates or renders obvious a “direct link,” as recited in all independent claims. EX2015, ¶62.

2. Raciborski requires DNS resolution and thus does not cure Verma’s “direct link” deficiency (Grounds 2, 4, 5).

Petitioner does not rely on Raciborski to supply either the “modified embedded resource” or the “direct link” limitation of any claim. Raciborski’s role is limited to the parameter-passing and API limitations. Pet., 33-43. To the extent Petitioner argues otherwise, Raciborski independently fails to disclose a “direct link” as properly construed. Raciborski’s system also delivers content objects using a CDN hostname—specifically a hostname including the “LLNW.com” domain name. EX1006, ¶¶55, 61, Table I, Table III. Like Verma, this hostname requires

DNS resolution; it does not serve as a DNS-bypassing direct link. EX2015, ¶63.

Thus, even assuming that Raciborski could be combined with Verma, the resulting combination would not produce a DNS-bypassing “direct link” because neither reference discloses or suggests returning a DNS-bypassing address as the host portion of a modified embedded resource URL. EX2015, ¶63. Adding Raciborski’s LLNW.com hostname URLs to Verma’s akamaized hostname URLs would simply yield a combined system that still depends entirely on conventional DNS resolution—the opposite of what the ’322 patent claims. EX2015, ¶63.

3. Lewin also requires DNS resolution and thus does not cure Verma’s “direct link” deficiency (Grounds 3, 4, 5).

Petitioner relies on Lewin for the “tag” limitation in Ground 3 and subsequent grounds incorporating Lewin. Pet., 43-62. But to the extent Petitioner argues otherwise, Lewin’s system is itself DNS-dependent, and any alleged combination of Verma and Lewin does not produce a DNS-bypassing direct link. EX2015, ¶64.

Lewin’s own disclosure confirms that DNS resolution is required. Lewin describes a sequence in which “an end user enters a URL into his or her browser, e.g. <http://www.cp.com/index.html>. The browser makes a DNS request to resolve www.cp.com and receives back an IP address for a given ‘edge’ server in the ICDN.” EX1007, ¶49. Lewin’s edge servers and the HTML resources they serve are thus reached only after the client performs a full public DNS lookup. *Id.*; *see*

also EX2015, ¶65. This architecture is the same DNS-dependent addressing architecture that the '322 patent's "direct link" is designed to bypass. EX2015, ¶65.

Moreover, the content fragments assembled through Lewin's ESI tag processing do not constitute DNS-bypassing resources. Lewin's ESI tags (e.g., <esi:include>) are assembly directives that instruct an edge server to incorporate additional HTML content fragments into a page. EX1007, ¶¶28, 30, 49. Those fragments are retrieved from the same DNS-resolved edge server infrastructure, and at no point in Lewin's workflow is an IP-literal or otherwise DNS-bypassing address generated. EX1007, ¶49; EX2015, ¶66. Even assuming that Lewin could be combined with Verma, the resulting system's URLs would remain hostname-based—as both references use CDN hostnames—and still require DNS. Neither reference, individually or combined, discloses a mechanism for generating a DNS-bypassing link as required by the '322 patent independent claims. EX2015, ¶66.

Accordingly, every reference relied on for the independent claims—Verma, Raciborski, and Lewin—employs hostname-based addressing that requires conventional DNS resolution. Therefore, the Petition fails to establish that any challenged claim is disclosed or rendered obvious by Verma, alone or in combination with Raciborski or Lewin.

B. The Petition fails to establish that Verma, alone or in combination with Lewin, renders obvious the “tag” limitations of independent claim 1 (Grounds 3-5).

Independent claim 1 recites, “receiving, by an origin system, a request from a client device for a content resource, the content resource including *at least one embedded resource with a tag*,” and “*upon identification of the tag*, using at least one delivery parameter to *obtain, by the origin system, a modified embedded resource* from a content delivery network that is separate from the origin system.” EX1001, 10:5-12 (elements [1a] and [1b]).

Petitioner fails to establish that Verma in combination with Lewin render obvious these limitations for three independent reasons. First, Lewin does not disclose an “embedded resource with a tag,” as required by independent claim 1. Second, Petitioner has not established a motivation to combine Verma and Lewin. Third, Petitioner has not established a reasonable expectation of success in achieving the combination. Further, even if the references could be combined, the resulting system would still lack a “direct link” for the reasons set forth in Section V.A, *supra*.

Therefore, Petitioner fails to show that claims 1-10 are unpatentable.

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

Petitioner acknowledges that “Verma does not expressly disclose its origin system receiving ‘a request from a client device for a content resource, the content

resource including at least one embedded resource with a tag,” instead relying on Lewin for the claimed “embedded resource with a tag.” Pet., 45-48. In particular, the Petition asserts that Lewin’s “‘URLs’ are embedded resources,” and these URLs are “‘specif[ied] by ‘tags’ like a ‘<esi:include> tag.’” Pet., 47 (citing EX1007, ¶¶28, 30, 49. But Lewin does not supply this missing “embedded resource with a tag” limitation. Unlike in the ’322 patent, Lewin’s purported “embedded resource” (e.g., a URL) does not have a tag; Lewin’s “tag” is not part of the “embedded resource” at all. EX2015, ¶70.

Specifically, claim 1 of the ’322 patent recites “at least one *embedded resource with a tag*.” EX1001, 10:6-8. The ’322 patent specification describes this structure, explaining that “[t]he *embedded resource* may *include a tag* indicating that a resource address compute engine 30 should be contacted.” EX1001, 4:32-34. The specification further illustrates the form of the claimed “embedded resource with a tag”: “http://\${CALLAPI:ADDRONLY}/path/resource.txt, or \${CALLAPI:F’ULLURL, path=\$ENV{PATH}, pop=\$POP}.” EX1001, 4:34-48. As illustrated, the tag—the “\${}” construct—is embedded inside or is the URL string itself (i.e., an “embedded resource *with a tag*). *Id.*; EX2015, ¶71. In other words, the tag occupies the position within the address where the CDN node address will go; the URL cannot be used as-is because the tag is a placeholder that must be resolved. EX2015, ¶71.

Lewin, by itself or combined with Verma, does not disclose the claimed “embedded resource with a tag.” EX1001, 10:6-8. Lewin’s ESI tags are *not* part of the purported “embedded resource” identified by the Petition; rather, they bear the inverse structural relationship to their URLs. As exemplified in Lewin’s Figure 6, the relevant structure is: “<esi:include src=us_ad.html/>,” where the “<esi:include>” element is an HTML document-level tag, and the URL (“us_ad.html”) is a value carried as an attribute (“src=”) within the tag. EX1007, FIG. 6, ¶¶28, 30. In Lewin, the URL is inside the tag—not the other way around. EX2015, ¶72. Thus, Lewin’s tag is not a marker within the URL indicating that the URL address needs to be computed and replaced; instead, it is merely an HTML assembly directive that happens to reference a URL as one of its parameters. EX2015, ¶72.

This inverted structure reflects a fundamentally different functional design. In the ’322 patent, the tag signals to the origin server that the URL itself is incomplete and must be resolved. EX1001, 4:40-48; EX2015, ¶73. In Lewin, the tag signals to the edge server that a content fragment should be fetched from the complete URL contained in the tag’s attribute and spliced into the HTML page. EX1007, ¶¶28-30, 49. The structural difference—tag-in-URL vs. URL-in-tag—reflects that functional difference. EX2015, ¶73. Accordingly, the combination of

Verma and Lewin does not teach an “embedded resource *with* a tag,” as claim 1 requires. EX2015, ¶73.

Additionally, claim 1 requires, “upon identification of the tag, ... obtain[ing], *by the origin system*, a modified embedded resource from a content delivery network that is separate from the origin system.” EX1001, 10:9-12. The “<esi:include>” tag relied upon by Petitioner, Pet., 47-48, is an assembly directive parsed by a CDN edge server—not by an origin server—to incorporate additional HTML content fragments into a page. EX1007, ¶¶28, 30, 49; EX2015, ¶74. Lewin expressly describes its mechanism as “server side include processing” that is “carried out on an edge server.” EX1007, ¶28. The ESI tags instruct the edge server to fetch and assemble pre-existing content blocks (e.g., advertisements, personalized greetings, navigation elements) into a container HTML page at the time of delivery. *Id.*, ¶¶28-30, 42, 49; EX2015, ¶74. Put simply, Lewin discloses HTML content assembly at an edge server, not address computation or replacement initiated by an origin server. EX2015, ¶74.

Therefore, Petitioner fails to demonstrate that the combination of Verma and Lewin renders obvious the “tag” limitations of claim 1.

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

Even if Lewin taught the claimed “tag” limitations of claim 1 (it does not), Petitioner does not establish that a POSITA would have been motivated to

combine Verma and Lewin in the manner proposed. EX2015, ¶¶76-82. Petitioner's stated motivation for the Verma-Lewin combination is that adding Lewin's ESI tag-parsing to Verma would give Verma's system "a uniform and efficient way to identify URLs" and "dynamic identification and execution of the URLs." Pet., 50-56; EX1003, ¶¶213-217, 225-235. Petitioner fails to establish a motivation to combine for three independent reasons: (1) Verma already discloses markup for identifying URLs, and Petitioner provides no reason why a POSITA would have sought to modify Verma's existing markup scheme; (2) Verma's content provider does not require markup tags to identify URLs; and (3) combining Lewin's ESI tag-processing architecture with Verma's system would not achieve the claimed invention. EX2015, ¶76.

First, adding Lewin's ESI tags to Verma's system to "identify URLs" would be purpose-defeating redundancy. As Petitioner recognizes, Verma already includes standard HTML tags (e.g., "a href" tags) to identify URLs. Pet., 51; EX1005, ¶29; EX2015, ¶77. The <a href> tag is the standard HTML mechanism for embedding and identifying a URL in a document—that is its precise function, and it performs it in Verma. EX2015, ¶77. Layering Lewin's ESI <esi:include src="[URL]"/> tags on top of a system that already uses <a href> for URL identification would not improve URL identification; it would add a second, structurally different tag system to accomplish the same URL-carrying function the

first system already performs. EX2015, ¶77. A POSITA would not introduce such redundancy. EX2015, ¶77; *Virtek Vision Int'l ULC v. Assembly Guidance Sys., Inc.*, 97 F.4th 882, 887 (Fed. Cir. 2024) (“[t]he mere fact that these possible arrangements existed in the prior art does not provide a reason that a skilled artisan would have substituted” one arrangement for the other).

Second, Verma’s system operates by allowing the client browser to request a content file directly—i.e., “by sending a URL 28 to a content provider 22” to process. EX1005, ¶25. Petitioner alleges that “[i]t would have been obvious for Verma’s ‘URL 28’ to be embedded in an HTML page” but provides no reason whatsoever to do so. Pet., 49-50. Indeed, it would make little sense to require Verma’s “browser 21” to construct an HTML page with URL 28 embedded within the page before sending a request to content provider 22. EX2015, ¶78; EX1005, ¶25. Verma’s content provider 22 does not require the presence of any tag to identify URL 28; URL 28 is already received directly from browser 21. EX1005, ¶25; EX2015, ¶78. Such operation would add unnecessary and undesired complexity to Verma’s system that can only be attributed to improper hindsight. EX2015, ¶78; *KSR Int’l Co. v. Teleflex Inc.*, 550 U.S. 398, 421 (2007) (“A factfinder should be aware, of course, of the distortion caused by hindsight bias and must be cautious of arguments reliant upon ex post reasoning.”); *Cheese Sys., Inc. v. Tetra Pak Cheese & Powder Sys., Inc.*, 725 F.3d 1341, 1352 (Fed. Cir.

2013) (“Obviousness cannot be based on the hindsight combination of components selectively culled from the prior art to fit the parameters of the patented invention.” (cleaned up); *DirectTV, LLC v. Qurio Holdings, Inc.*, IPR2015-02006, Paper 6 at 13-14 (PTAB Apr. 4, 2016) (“Ex post reasoning” and “hindsight” are forbidden in the obviousness analysis).

Third, combining Lewin’s ESI tag-processing architecture with Verma’s system would not achieve the claimed invention. EX2015, ¶79. Lewin’s ESI tag-processing architecture is fundamentally incompatible with the operational sequence element [1b] requires, which is “upon identification of the tag, using at least one delivery parameter to obtain, by the origin system, a modified embedded resource from a content delivery network.” EX1001, 10:8-12; EX2015, ¶79. The information and computation flow runs from the origin server to CDN: the origin server identifies the tag, invokes the CDN’s API with delivery parameters, and receives back a modified embedded resource. EX1001, 4:49-52, 6:24-30; EX2015, ¶79.

In Lewin, that processing direction is reversed. The origin system is the entity that creates the HTML template containing ESI tags by placing <esi:include> directives into the page. EX1007, ¶28. But the origin system does not identify or process those tags. EX2015, ¶80. Tag identification and processing occurs at the CDN “edge server,” which parses the template “looking for tags that

specify dynamic assembly instructions” and fetches the referenced content fragments from the origin to assemble the final page. EX1007, ¶49. Contrary to the claimed approach, the information flow runs from CDN to origin: the CDN edge identifies the ESI tag, contacts the origin to retrieve content fragments, and assembles the page locally. EX2015, ¶80. Thus, producing a system that satisfies element [1b] using Lewin’s ESI architecture would require redesigning Lewin’s techniques to account for the opposite processing direction. That is not a routine combination of known elements but is rather an architectural redesign that a POSITA would not have sought to pursue predictably achieved. EX2015, ¶81; *Plas-Pak Indus., Inc. v. Sulzer Mixpac AG*, 600 F. App’x 755, 758 (Fed. Cir. 2015) (“[C]ombinations that change the ‘basic principles under which the [prior art] was designed to operate,’ ... may fail to support a conclusion of obviousness.” (citing *In re Ratti*, 270 F.2d 810, 813 (C.C.P.A. 1959))).

For at least these reasons, Petitioner fails to establish a motivation to combine Verma and Lewin.

3. Petitioner fails to explain how tag identification in the combined system triggers the origin system’s use of delivery parameters to obtain a modified embedded resource.

Element [1b] recites “upon identification of the tag, using at least one delivery parameter to obtain, by the origin system, a modified embedded resource” and imposes a specific functional sequence: (1) the origin system identifies a tag in

the embedded resource; (2) that identification triggers the origin system to use delivery parameters; and (3) the origin system uses those parameters to obtain a modified embedded resource from the CDN. EX1001, 10:9-12. However, the combination Petitioner proposes—Verma for the URL-prepend mechanism and Lewin for the concept of tag identification—leaves the connective tissue of this sequence entirely unexplained. EX2015, ¶83.

For the “upon identification of the tag” limitation, Petitioner concedes Verma does not expressly disclose this element and turns to Lewin. Pet., 54. But Petitioner does nothing more than assert that incorporating Lewin’s teachings of parsing an HTML page, a Lewin-modified Verma would perform an “identification of [a] tag that initiated dynamic assembly, like incorporation of content ‘into the final HTML page’ and obtaining a modified embedded resource, like URLs on the HTML page.” Pet., 55; EX2015, ¶84. This is the entirety of Petitioner’s explanation for how tag identification connects to Verma’s URL-prepend process. This is not an explanation but rather a conclusory assertion.

Dr. Mowry’s testimony on this precise point is no more illuminating. Dr. Mowry similarly acknowledges Verma does not expressly disclose obtaining a modified embedded resource “based upon identification of the tag.” EX1003, ¶233. To fill this gap, he turns to Lewin’s description of a server that “‘parses’ an HTML page ‘looking for tags’” and “‘evaluate[s]” directives “‘placed into HTML pages.’”

EX1003, ¶234 (citing EX1007, [0028], [0049]). From this, Dr. Mowry concludes that “a POSITA would have understood the identification of the tag initiated the dynamic assembly, like ‘incorporat[ion of content] into the final HTML page’ and obtaining a modified embedded resource.” *Id.* But this conclusion elides the central question: in the proposed combination, how does tag identification by the origin system cause the origin system to use Verma’s alleged delivery parameters and invoke Verma’s URL-prepend mechanism? Dr. Mowry simply does not say. EX2015, ¶84.

Nowhere in the Petition or Dr. Mowry’s declaration is there any actual explanation of how those two pieces connect in the combined system to satisfy the “upon identification . . . obtain” causation sequence the claim requires. EX2015, ¶85. In Verma, the origin system sends request 34 to the broker and receives a prepended URL in return—a process wholly untethered to tag identification, because Verma’s system does not involve parsing any HTML for tags at all. EX1005, ¶¶25-26; EX2015, ¶85. And in Lewin, tag parsing is performed by a server in the assembly layer to fetch and incorporate content fragments, not by an origin system to obtain a CDN-modified URL using client delivery parameters. EX1007, ¶¶28, 49; EX2015, ¶85. Neither reference, nor the two in combination as described, answers the question of how identifying a tag in the embedded resource

causes the origin system to invoke Verma’s delivery-parameter-driven URL-rewriting process.

This type of gap—leaving the Board to infer how the claimed functional causation would arise from the combination—is precisely what the Federal Circuit has held is insufficient. *See TQ Delta, LLC v. Cisco Systems, Inc.*, 942 F.3d 1352, 1359 (Fed. Cir. 2019). Dr. Mowry’s testimony on this point “merely repeat[s]” the Petition’s assertion and does not supply the missing reasoning. *Xerox Corp. v. Bytemark, Inc.*, IPR2022-00624, Paper 9 at 15 (PTAB Aug. 24, 2022). It is accordingly entitled to little weight. *See Intelligent Bio-Sys., Inc. v. Illumina Cambridge Ltd.*, 821 F.3d 1359, 1369 (Fed. Cir. 2016) (petitioner bears the burden to explain with particularity how the combination renders the claim obvious under 37 C.F.R. § 42.104(b)(4)).

The same evidentiary gap that undermines Petitioner’s teaching of element [1b] independently establishes the absence of a reasonable expectation of success. A showing of reasonable expectation of success requires more than proof that each component of a combination existed — it requires a showing that a POSITA would have had a reasonable basis to believe the specific combination would work as claimed. *Id.* at 1367-68. Here, neither Petitioner nor Dr. Mowry has identified 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 “upon identification of the tag, using at least one delivery parameter to obtain” sequence. Without that explanation, there is no basis to conclude a POSITA would have reasonably expected the combination to succeed — only that the components exist in isolation. EX2015, ¶86. That is legally insufficient. *Medichem, S.A. v. Rolabo, S.L.*, 437 F.3d 1157, 1165 (Fed. Cir. 2006); *see also Virtek Vision*, 97 F.4th at 886 (Fed. Cir. 2024). Reasonable expectation of success requires a showing specific to the proposed combination, not just the availability of components among “many possible choices.” *Medichem*, 437 F.3d at 1165.

For these additional reasons, Petitioner has not established that the combination of Verma and Lewin renders obvious the “tag” limitations of claim 1.

C. The Petition fails to establish that the cited art discloses or renders obvious the claimed “application programming interface (API)” (Grounds 1A/1B, 2, 4; Claims 11 and 15).

Each of claims 11 and 15 recites “providing, at a content delivery network, an application programming interface (API)” and “at the API, receiving, from an origin system that is separate from the content delivery network, an embedded resource and at least one parameter associated with the embedded resource.” EX1001, 10:63-67 (Claim 11), 12:6-11 (Claim 15). Dependent claims 2 and 9 also similarly require the origin system to pass “a location identifier” or “information

based on the client IP address” to “an application programming interface (API) of the content delivery network.” EX1001, 10:23-27 (Claim 2), 10:52-55 (Claim 9).

The Petition relies solely on Verma for the recited “application programming interface (API)” across all Grounds. Pet., 11-12. Although Grounds 2 and 4 further rely on Raciborski and Lewin for certain limitations, Petitioner does not rely on either reference for the API. Pet., 34, 37-39, 54, 67-68, 77. For the reasons below, Petitioner does not establish that Verma’s broker provides the claimed “API,” and this deficiency is fatal to claims 2, 9, and 11-15.

1. Claims 2, 9, and 11-15 require an exposed “API” to receive requests, regardless of whether the API includes additional functionality.

In its Institution Decision, the Board’s preliminarily found that the claimed “API need not be separate from the engine executing on the server.” DI, 10 (citing EX1001, 7:22-25). Based on this preliminary finding, the Board determined on the current record that “Verma’s broker fulfills Dr. Russ’s characterization of an API acting as an interface that allows the client to communicate with an application executing on the server (that which provides availability information to the broker).” DI, 13-14. The Board’s analysis respectfully reads the term “API” out of the claims. First, the Board’s reliance on the ’322 patent specification’s description that “the API may include the functionality of the engine and not necessarily be considered a discrete mechanism from the computing engine functions,” Paper 17,

10 (citing EX1001, 7:22-25), is misplaced. EX2015, ¶90. Under proper interpretation, the claims do not encompass an embodiment in which the recited “API” is collapsed with the underlying engine executing on the server. Second, regardless of whether the claimed “API” encompasses the underlying engine, the requirement of providing a defined application programming interface still exists. Neither Verma nor the Petition identifies one. EX2015, ¶91.

a. The Board preliminarily determined that “because the ’322 patent states that the API ‘may include’ the functionality of the engine, Patent Owner’s example does not show that the API and engine must always be entirely separate components.” DI, 10 (referencing EX1001, 7:22-25); *see also* Pet., 12; Paper 15, 3-4 (Petitioner relying on the same passage). EX2015, ¶92. This passage, however, does not support such a broad reading of the claims. Instead, it describes an unclaimed, alternative embodiment, made clear from an examination of the full passage:

In an alternative implementation, which may also involve an API and calling the same in order to resolve an optimal node to serve a request, the process of contacting the API and constructing the optimal network address is instead handled by the CDN and does not necessarily involve a customer origin server or other devices outside the CDN. While embodiments are discussed herein as involving an API, it should be noted that an API is a convenient and efficient way for disparate systems to uniformly and predictably pass information to the

computing engine and for the computing engine to return information. However, other forms of communicating with and passing information to the computing engine are also possible. Moreover, the API may include the functionality of the engine and not necessarily be considered a discrete mechanism from the computing engine functions.

EX1001, 7:10-25. Significantly, the passage expressly introduces “an alternative implementation.” EX1001, 7:10. The specification thus signals that what follows—including the “not necessarily discrete” language—describes a different embodiment from the primary API architecture disclosed elsewhere in the specification. EX2015, ¶92.

That this passage describes an alternative implementation is further underscored by the description of this implementation “not necessarily involv[ing] a customer origin server or other devices outside the CDN.” EX1001, 7:13-15. That is the opposite of what claims 11 and 15 recite. EX2015, ¶93. Those claims expressly recite “at the API, receiving, from an origin system that is separate from the content delivery network, an embedded resource and at least one parameter associated with the embedded resource.” EX1001, 10:63-67 (Claim 11), 12:6-11 (Claim 15). A POSITA reading the specification would have thus recognized that the “not necessarily discrete” language in this passage describes an alternative embodiment in which the CDN handles address resolution internally—not the

claimed embodiment in which the origin server calls an external API exposed by the CDN. EX2015, ¶93.

The claimed API is instead described in the specification's primary embodiment, which is depicted in Figures 1 through 3 and described in detail. *See* EX1001, 3:6-7:9; EX2015, ¶94. In that embodiment, the API 26 and the computing engine 30 are shown and described as distinct components: "the API 26 provides the interface to a computing engine 30 that uses the information, passed to the engine through the API." EX1001, 6:24-30. The origin server affirmatively "calls an API in order to communicate with the compute engine 30 to format the address for the embedded resource." EX1001, 4:49-52. Figure 3 illustrates this architecture, with a distinct operation for the compute engine receiving inputs through the API (Operation 310) and separate operations for the compute engine identifying nodes and the API returning a network address to the origin server (Operations 320-330). EX1001, FIG. 3, 6:24-53. The API in the primary embodiment is thus a defined, distinct interface layer through which the external origin server invokes the CDN's computational functionality—and this is the API that the claims capture. EX2015, ¶94.

The claim language itself confirms this reading. As noted, the claims require the API to "receiv[e], from an origin system that is separate from the content delivery network," an embedded resource and parameters, an act of receiving

external input that presupposes the API is an interface accessible to an external caller. EX1001, 10:63-67. Notably, the claims then separately recite “generating a modified embedded resource” *without attributing that step to the API*. EX1001, 10:68-11:2 (Claim 11), 12:12-14 (Claim 15). This distinction is consistent with the primary embodiment’s architecture where the API receives the inputs, and the compute engine generates the output. EX2015, ¶95. 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; the claims could simply recite the CDN generating the modified embedded resource. EX2015, ¶95. That the claims call out a distinct “API” as the receiving mechanism confirms that the API is a structurally meaningful interface component, not merely a label for whatever internal CDN process happens to participate in content delivery. *See Intel Corp. v. Qualcomm Inc.*, 21 F.4th 801, 810 (Fed. Cir. 2021) (“It is highly disfavored to construe terms in a way that renders them void, meaningless, or superfluous.”).

b. Regardless of whether the ’322 patent’s claimed “API” encompasses the underlying engine, the requirement of providing a defined application programming interface still exists and should not be read out of the claims. T EX2015, ¶96. The specification’s description that the API may “not necessarily be considered a discrete mechanism from the computing engine functions” advises

only that the API may be architecturally integrated with the engine's functionality—for instance, implemented in the same software module or server. But it does not eliminate the requirement that the API provide an interface through which an external caller invokes computational functionality. In other words, the '322 patent specification permits implementation flexibility at the server back end while preserving the API's functional role. EX2015, ¶96. There must still be, however, an exposed API for receiving requests, regardless of whether the API is considered to include processing logic performed after receiving a request. EX2015, ¶96.

The differences between independent claim 1 and independent claims 11 and 15 further enforce this reading. In particular, claim 1 does not recite an API while claims 11 and 15 do. Claim 1 thus encompasses broader forms of communication between the “origin server” and the “content delivery network” aside from calling an exposed API. EX1001, 10:5-18; *see also id.*, 3:9-14 (disclosing either “calling an API *or otherwise querying a CDN.*”). To interpret claims 1, 11, and 15 to all encompass the same embodiments would render the “API” recited in claims 1, 11, and 15 meaningless. *Curtiss-Wright Flow Control Corp. v. Velan, Inc.*, 438 F.3d 1374, 1381 (Fed. Cir. 2006) (“[C]laim differentiation takes on relevance in the context of a claim construction that would render additional, or different, language in another independent claim superfluous.”); *Karlin Tech., Inc. v. Surgical*

Dynamics, Inc., 177 F.3d 968, 971-72 (Fed. Cir. 1999) (“[D]ifferent words or phrases used in separate claims are presumed to indicate that the claims have different meanings and scope.”); EX2015, ¶97.

As discussed below, Petitioner points to nothing in Verma aside from generic communication between two servers, effectively arguing that any device that receives communication from another device would automatically satisfy the claimed “API” limitations. EX2015, ¶98. That interpretation reads the term “API” out of the claims and is not sufficient to render claims 2, 9, and 11-15 obvious.

2. Petitioner has not shown that Verma’s broker provides the claimed “API.”

Petitioner contends that “a POSITA would have understood Verma’s ‘broker’ is an API because it performs the same logical functions as the API in the ’322 patent.” Pet., 12 (citing EX1001, 4:49-52, 7:16-25); *see also* EX1003, ¶87. That assertion misses the point. Petitioner’s analysis establishes nothing more than Verma’s “broker” acts as an intermediary between Verma’s “content provider” and “cache servers.” Pet., 11-12. That Verma’s broker receives and transmits information does not mean that this communication is performed through an exposed API. EX2015, ¶99.

Verma’s broker is an internal CDN management and scheduling component. EX2015, ¶100. It “manages at least one, and preferably, a set of cache servers,” sends “availability inquiries” to cache servers to “find out whether they currently

store the requested” content file, and “schedule[s] the distribution of content files to edge servers.” EX1005, ¶¶22, 25. When content is not immediately available, the broker “negotiate[s] with the content delivery network” to “determine the start time, end time, and the cache server where the content file will next be available.” EX1005, ¶38. These are the functions of a CDN-internal scheduling and routing component, not of an interface through which an external origin server invokes CDN computational functionality. EX2015, ¶100.

Critically, Verma’s content provider does not disclose that its content provider invokes the broker through a defined programmatic interface with structured inputs and outputs. EX2015, ¶101. Verma merely describes the content provider “send[ing] a request 34 for information” to the broker, and the broker “provid[ing] content current and/or future availability information” back to the content provider. EX1005, ¶¶25-26. Verma does not disclose or discuss the structure of request 34—including what specific information the content provider passes to the broker. *See* EX1005, ¶¶25-26; Paper 6 at 25; EX2016, 36:15-37:12 (Petitioner’s expert agreeing that “[t]here is not a detailed description of a specific syntax for -- or format” of “the request that’s sent from content provider 22 to broker 25.”). No external system makes an API call to Verma’s broker to generate a modified embedded resource. EX2015, ¶101. Rather, the broker mediates content delivery requests through Verma’s internal pipeline.

The distinction between Verma's broker and the claimed API is not merely semantic. In the '322 patent's primary embodiment, the API exists precisely to provide the origin server with a defined interface that facilitates communication with a separate compute engine in the CDN. EX1001, 6:24-30 ("the API 26 provides the interface to a computing engine 30"); EX2015, ¶102. If the origin server could simply forward a request into the CDN's internal pipeline, as Verma's content provider does with request 34 (EX1005, ¶25), there would be no need for an API at all. EX2015, ¶102.

Petitioner's failure to explain how Verma's broker provides an API is further underscored by the '322 patent's distinction between "calling an API *or otherwise querying the CDN.*" EX1001, 3:9-14; EX2015, ¶103. In other words, the '322 patent specification contemplates communication with CDN components *without* calling an API. EX2015, ¶103. That Verma's broker receives requests from Verma's content provider does not resolve the question of whether those requests are received by calling an API. EX2015, ¶¶103, 107 (explaining that "any HTTP call to a server does not suffice to establish use of an API"). And neither the Petition's nor Dr. Mowry's analysis address that question. *See* Pet., 11-12; EX1003, ¶87. Tellingly, Dr. Mowry refused to answer during deposition whether he had even "consider[ed] any distinction between calling an API or otherwise

querying the CDN.” EX2016, 92:19-96:20; *see also id.*, 102:20-105:2, 119:21-120:16.

Accordingly, the record evidence fails to establish that Verma’s broker provides the claimed “API” of claims 2, 9, and 11-15. These claims should be found patentable.

3. Dr. Mowry’s testimony and the Board’s preliminary reliance on Dr. Russ’s general API definition do not fill the gaps in Verma.

Dr. Mowry attests that Verma’s broker “performs the same logical functions as the API in the ’322 patent” because it communicates between the content provider and the CDN. EX1003, ¶87. But Dr. Mowry’s analysis consists of describing what Verma’s broker does in general terms—receiving a forwarded request, selecting a cache server, returning a rewritten URL—and then labeling it an “API.” EX1003, ¶¶87-88. This approach conflates any intermediary component in a data flow with a programmatic interface. EX2015, ¶104. A router sits between a client and a server and a load balancer sits between a client and a pool of servers, yet neither is an API. The claims require an actual interface that receives defined inputs from an external origin system and generates a defined output. EX2015, ¶104. Dr. Mowry does not identify where Verma discloses this specific input-output relationship at the broker, because Verma does not disclose it. *Id.*

The Board’s preliminary finding on this limitation also relied on Dr. Russ’s general characterization of an API as “the bridge establishing the connection between” a client and server, concluding that “this function of Verna’s broker fulfills Dr. Russ’s characterization of an API.” Paper 17 at 13. Patent Owner respectfully submits that this finding, while understandable on the preliminary record, applied Dr. Russ’s background definition at too high a level of abstraction. Dr. Russ’s description of an API as a “bridge” was a general explanation of what an API is as a concept in the field of computer science. EX2010, ¶50; EX2015, ¶105. It was not intended to define the metes and bounds of the API limitation as used in the claims of the ’322 patent. EX2015, ¶105.

As Dr. Russ explains, a POSITA would have understood that an “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. Defining characteristics include: (1) the interface is accessible to an external caller (here, the origin server); (2) it accepts structured inputs through defined parameters; and (3) it returns a defined output in response. EX2015, ¶¶106-107. This is precisely the architecture the ’322 patent describes: the origin server “calls an API” (EX1001, 4:49-52) using defined inputs—the embedded resource, client IP address, popularity, and geographic information—and receives back a defined

output, the “network address of identified node.” EX1001, FIG. 3, 6:50-7:1;
EX2015, ¶106.

By contrast, merely forwarding a request through an internal CDN pipeline —without a defined interface contract, defined parameters, or a defined output — is not an API call. EX2015, ¶¶107-108. A router receives packets and forwards them, and a load balancer receives requests and distributes them, yet those intermediary functions do not make them APIs. EX2015, ¶108. The distinction is whether the external caller invokes a specific, defined interface with a specified input-output contract, or whether it simply sends a message into a system’s internal pipeline. EX2015, ¶108. Verma’s broker receives a “request 34” from the content provider (EX1005, ¶¶25-26), but Verma is entirely silent on the structure of request 34—what specific parameters the content provider passes, what interface definition (if any) governs the exchange, and what defined output the broker returns. EX2015, ¶108; EX2016, 36:15-37:12. Verma discloses no defined programmatic interface, nor does Petitioner point to one. EX2015, ¶108.

VI. CONCLUSION

For the reasons set forth above, claims 1-15 should be confirmed as patentable over the asserted Grounds upon which this IPR was instituted.

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

Date: March 5, 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 Response complies with the type-volume limitation of 14,000 words, comprising 10,671 words, excluding the parts exempted by 37 C.F.R. § 42.24(a)(1).
2. This Patent Owner Response 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: March 5, 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 RESPONSE** and associated Exhibits 2015-2026 were served in their entireties on March 5, 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: March 5, 2026

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