

Google LLC
v.
Sandpiper CDN, LLC

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



Patent Owner's Demonstratives
September 1, 2026

Sandpiper EX2027
Google v. Sandpiper
IPR2025-01010 1

Sandpiper CDN – History

History



Sandpiper Networks built and launched the world's first CDN in the 1990s. The company's early customers included the L.A. Times, which used Sandpiper's CDN to host the report of Independent Counsel Ken Starr on his investigation of President Bill Clinton.

In December 1999, Sandpiper Networks was acquired by Digital Island in a deal that valued Sandpiper Networks at over \$1 billion.

Following a series of additional mergers and acquisitions, the Sandpiper CDN business was acquired by Level 3 Communications.

The Sandpiper team continued developing Sandpiper's networking technology with significant investments by Level 3. Eventually rebranded as Lumen, its customers included Netflix and Disney, which used Sandpiper's networking technology to support their large video streaming platforms.

In late 2023 Lumen began to spin off its content delivery business, and Sandpiper's technology, software, and intellectual property is now owned and managed by Sandpiper CDN. Sandpiper CDN is co-owned by Andrew Swart, who was the CEO and co-founder of Sandpiper Networks.

Instituted Grounds

Claims	Ground #	35 U.S.C. §	References
15	1A	102	Verma
11, 12, 14, 15	1B	103	Verma
11, 12, 14, 15	2	103	Verma and Raciborski
1, 8, 10, 13	3	103	Verma and Lewin
2, 4, 5, 7, 9, 13	4	103	Verma, Raciborski, and Lewin
3, 6	5	103	Verma, Raciborski, Lewin, and Draï

Agenda

- ▶ **The '322 Patent**
- ▶ “a direct link to a node within the content delivery network”
 - ▶ Independent Claims 1, 11, and 15 / All Grounds
- ▶ The “tag” limitations
 - ▶ Independent Claim 1 / Grounds 3-5
- ▶ The “application programming interface (API)” limitations (LEAP)
 - ▶ Claims 2, 9, 11, and 15

The '322 Patent, Independent Claim 1

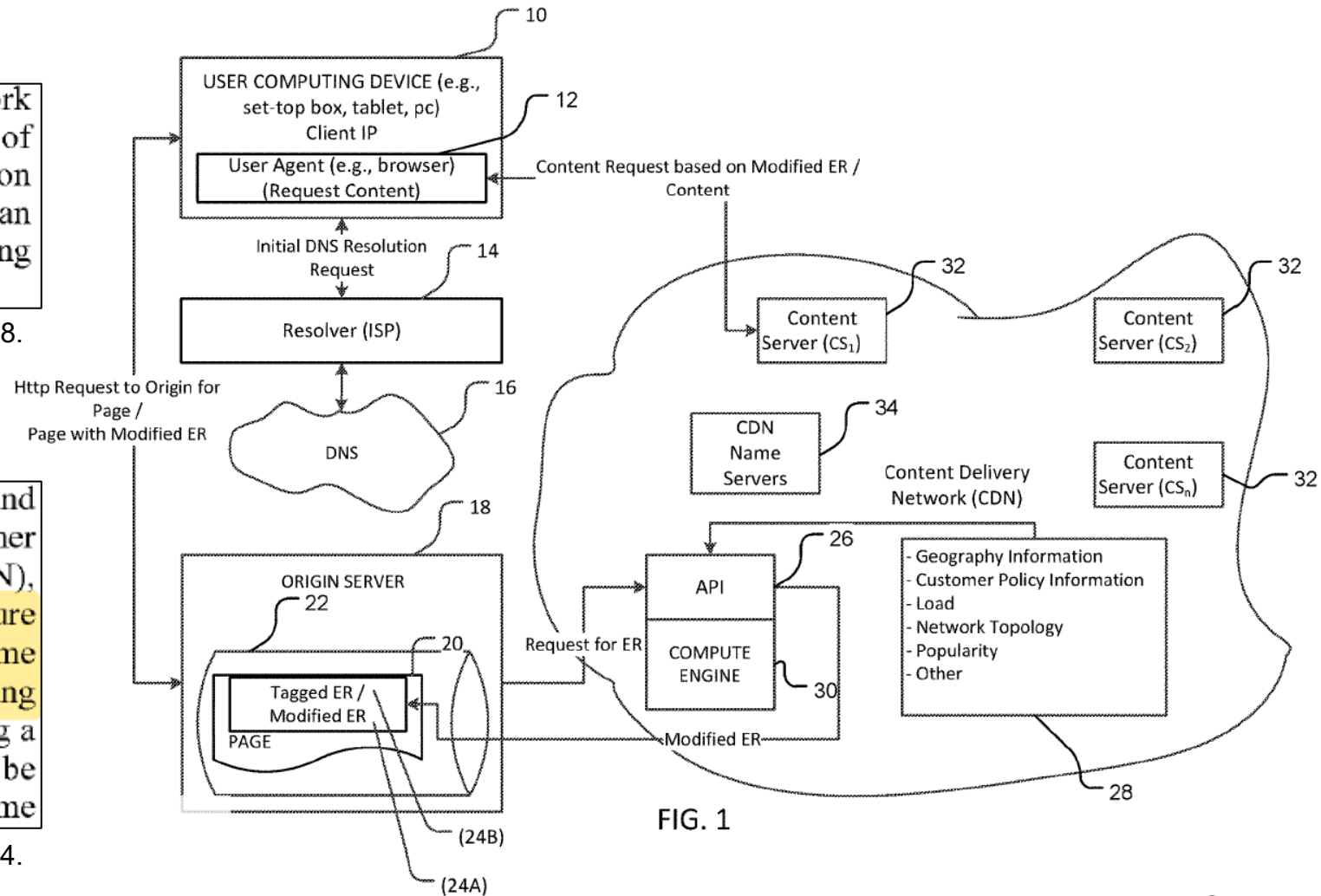
'322 Patent

Aspects of the present disclosure relate to network address resolution, and particularly involve introduction of an application programming interface into the resolution stream to embed link resolution information within an HTML document or other content returned to a requesting client.

EX1001, 1:13-18.

Aspects of the present disclosure involve systems and methods for selecting an edge server, rack, cluster or other device, which may be in a content delivery network (CDN), to serve content to a client. Aspects of the present disclosure 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. The API call may include a name

EX1001, 3:6-14.



EX1001, FIG. 1.

The '322 Patent, Independent Claim 1

1. A content delivery method comprising:
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;
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; and
delivering, by the origin system, the content resource to the client device with the modified embedded resource, 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, Claim 1.

The '322 Patent, Independent Claim 11

11. A content delivery network comprising:
at least one processor in communication with a non-transitory computer readable medium including computer readable instructions for:
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 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, Claim 11.

The '322 Patent, Independent Claim 15

15. 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, Claim 1.

Agenda

- ▶ The '322 Patent
- ▶ **“a direct link to a node within the content delivery network”**
 - ▶ Independent Claims 1, 11, and 15 / All Grounds
- ▶ The “tag” limitations
 - ▶ Independent Claim 1 / Grounds 3-5
- ▶ The “application programming interface (API)” limitations (LEAP)
 - ▶ Claims 2, 9, 11, and 15

The '322 Patent – “direct link”

'322 Patent, Claim 1

delivering, by the origin system, the content resource to the client device with the modified embedded resource, 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, Claim 1.

'322 Patent, 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;

EX1001, Claim 11.

'322 Patent, 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;

EX1001, Claim 15.

The '322 Patent – “direct link”

The '322 patent explains that a “direct link” is one that enables the client to contact a CDN node without performing conventional DNS resolution of a hostname.

Patent Owner's Construction

A network address that enables a client device to bypass conventional DNS resolution of a hostname

POR, 11-12; PO Sur-reply, 3.

Petitioner's Construction

A link pointing to a specific server

Reply, 1.

The '322 Patent – “direct link”

The '322 patent explains that a “direct link” is one that enables the client to contact a CDN node without performing conventional DNS resolution of a hostname.

'322 Patent

to serve content to a client. Aspects of the present disclosure 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. The API call may include a name

EX1001, 3:9-14.

FIG. 1 is a network 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, 2:45-49.

FIG. 4 is a network diagram and message flow related to generating an embedded resource from within a CDN for inclusion in a network asset, such as an HTML document, the embedded resource providing a connection to a node of the CDN bypassing DNS, in one example;

EX1001, 2:58-62.

The '322 Patent – “direct link”

The '322 patent explains that a “direct link” is one that enables the client to contact a CDN node without performing conventional DNS resolution of a hostname.

'322 Patent

request for the embedded resource. Note, 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. Regardless, the HTML document 20

EX1001, 5:29-32.

and return the modified document to the origin server. Thus, when the client browser displays the Web page, for example, the modified embedded resource (e.g., link) may include an absolute URL for a node (e.g., edge server(s)) within the CDN that can deliver the content. When the user selects the link, the browser will directly contact the node for the content. In the case of a hostname, 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 such that the CDN may perform any

EX1001, 6:56-65.

the CDN able to process the request. It may be the case that the hostname will also include a reference to a node capable of serving the request as determined by the API. In such a case, the hostname references a CDN location as is closely tailored to a location for serving the request but allows for some level of fail over, load balancing, or the like, relative to an absolute URL. In such a case, the hostname may be mapped to a VIP associated with various machines capable of serving the request.

EX1001, 7:1-9.

The '322 Patent – “direct link”

Petitioner Reply

2012) (citation omitted). Claim 11 recites a “modified embedded resource providing a direct link to a node,” but claim 14, which depends from claim 11, recites “the modified embedded resource includes at least one of an IP address.” EX1001, 11:1-4, 11:17-12:4. Claim 11’s use of “direct link” is broad enough to include *not only* claim 14’s IP address, *but also* other links, like hostnames, that might require DNS resolution to make a one-to-one translation to an IP address, where that hostname is bound to a specific node. This attempt to read claim 14’s “IP address” into claim 11 violates the doctrine of claim differentiation and should be rejected. *InterDigital*.

Reply, 3-4.

'322 Patent, 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;

EX1001, Claim 11.

This “direct link” points to a specific server for content retrieval. The claim language does not narrowly define the form a “direct link” must take. POR, 12-14. It could be a hostname, IP address, virtual IP address, or the like. *Nor does this claim language*

Reply, 1-2.

'322 Patent, Claim 14

14. The content delivery network of claim 11 wherein the modified embedded resource includes at least one of an IP address to facilitate communication with the node to obtain the content associated with the embedded resource, a virtual IP address to facilitate communication with the node to obtain the content associated with the embedded resource, and one or more absolute uniform resource locators to facilitate communication with the node to obtain the content associated with the embedded resource.

EX1001, Claim 14.

The '322 Patent – “direct link”

Implementing a “direct link” with a hostname enables the CDN to be contacted directly, “bypassing any other DNS processing.”

'322 Patent

CDN that can deliver the content. When the user selects the link, the browser will directly contact the node for the content. In the case of a hostname, 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 such that the CDN may perform any form of intelligent traffic management (ITM) or other forms of edge selection and return an IP address of a node within the CDN able to process the request. It may be the case that

EX1001, 6:60-7:1.

the CDN able to process the request. It may be the case that the hostname will also include a reference to a node capable of serving the request as determined by the API. In such a case, the hostname references a CDN location as is closely tailored to a location for serving the request but allows for some level of fail over, load balancing, or the like, relative to an absolute URL. In such a case, the hostname may be mapped to a VIP associated with various machines capable of serving the request.

EX1001, 7:1-9.

The '322 Patent – “direct link”

District court infringement contentions do not redefine “direct link” or suggest that URLs requiring conventional DNS would satisfy the claims.

```
"streamingData": {
  "expiresInSeconds": "21540",
  "formats": [{
    "itag": 18,
    "url": "https://rr3---sn-qxo7rn7y.googlevideo.com/videoplayback?expire=1732325796\u0000",
    "mimeType": "video/mp4; codecs=\"avc1.42001E, mp4a.40.2\"",
    "bitrate": 342693,
    "width": 640,
    "height": 360,
    "lastModified": "1697929619917530",
    "quality": "medium",
    "fps": 30,
    "qualityLabel": "360p",
    "projectionType": "RECTANGULAR",
    "audioQuality": "AUDIO_QUALITY_LOW",
    "approxDurationMs": "1575195",
    "audioSampleRate": "44100",
    "audioChannels": 2
  }],
}
```

Excerpt from HTML code containing reference to the CDN node on which the video is stored.

Based on request from network separate from the previous example.

<https://www.youtube.com/watch?v=Tx69m9Jb0EY>

In further support, Google provides webpages with embedded videos. In doing so, Google further generates a modified embedded resource providing a direct link (e.g., a link pointing to specific cache server), as indicated below.

EX1019, 7.

The '322 Patent – “direct link”

The arguments made in the prosecution history similarly focus on DNS bypassing.

Traditionally, an origin server might modify links to such content in the web page to point to a name server for the CDN. When the user then clicks on such links in the web page, the client is directed to the name server, which then provides the client the address of a particular node within the CDN from which to retrieve the content. In examples of the present application, however, in response to a request for a web page or other resource, the origin server contacts an API of the CDN to obtain modified embedded resources that include a direct link to a particular node within the CDN for the client to obtain the additional content. Among other things, this allows the client to bypass the name server at the CDN when requesting the content at the CDN. This is generally illustrated, *inter alia*, in Fig. 1 of the present application, where the client 10 contacts the origin server 18, which contacts the API 26 of the CDN to obtain modified embedded resources. The origin server provides the modified embedded resources to the client. The client can then directly access a content server 32 in the CDN, while bypassing the CDN name servers 34:

EX1002, 69.

Verma does not render obvious the “direct link” limitations

Petitioner argues only that Verma satisfies the “direct link” limitations under its own construction—not under Patent Owner’s construction.

Petitioner Reply

As the Petition explains, Verma discloses a prepended URL that “provides a direct link to a CDN node ..., as it ‘comprises a link to a content server or a cache server[,]’ allowing a client to directly contact the node” to receive content. Pet., 23-24; EX1003, ¶¶119-121. No more is required. *Supra* § II.A.

Reply, 11.

Patent Owner contends this mapping “equates ‘direct link’ with any URL that, when followed, ultimately reaches a particular cache server.” POR, 22. Not so. Verma’s URL “provides a direct path bound to a single cache server node,” which is not just any URL that may reach a particular cache server—it is a URL *directly pointing to only one server*. Pet., 23-24. Verma’s URL precludes DNS from selecting the node because the specific node is selected and returned to the client before the client contacts any DNS. Verma’s URL is also unlike the “translation code”

Reply, 11.

Verma does not render obvious the “direct link” limitations

Verma discloses only “Akamai-ed” hostname-based URLs that rely on conventional DNS hierarchical processing to resolve the altered hostname.

Verma

[0013] The present invention enables extending request routing capability to cache servers, which can hold content at a future time period. For example, Akamai’s request routing 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 in the content delivery network. In accordance with an aspect of the invention, an “Akamai-ed” URL is a link that will be available during a future time period.

EX1005, ¶13.

[0026] The broker 25 provides content current and/or future availability information to Movies-Inc.Com content provider 22 where the content file URLs are prepended (Akamai-ed) for redirecting to cache servers. Additionally, infor-

EX1005, ¶26.

Petitioner’s Expert

locators, as claimed. As discussed above for [15c], Verma discloses a modified embedded resource. *Supra* § X.A.4 (Grounds 1A/1B, [15c]). For instance, Verma discloses several prepended URLs (modified embedded resources) including:

- http://location1/contentURL
- http://location2/contentURL, and
- http://local_cache_server/contentURL.

EX-1005, [0013], [0029], [0031]. Each of these prepended URLs is also an absolute URL because each provides a complete, absolute path to the single relevant cache server (e.g., “cache server 23”/“24”). *Id.*, [0027], [0037], FIG. 2. A POSITA would have understood that each of these uniform resource locators includes a protocol, “http”; a hostname for a cache server, “location1,” “location2,” or “local_cache_server”; and a path to content, “contentURL.” *Id.*, [0013], [0029],

EX1003, ¶151.

Verma does not render obvious the “direct link” limitations

Verma discloses only “Akamai-ed” hostname-based URLs that rely on conventional DNS hierarchical processing to resolve the altered hostname.

Petitioner’s Expert

answer. The first time the client tries to access now that altered host name that points to a Akamai-ed cache server, it still needs to obtain the mapping from the host name to the cache server's IP address, right?

MR. FELICIELLO: Objection. Form.

A When a URL includes a host name, it needs to be mapped to an IP address. But in this case, where the content delivery network has chosen the host name to point to a specific URL, it will always map to that specific URL -- sorry, that specific IP address. And it's been chosen to

EX2016, 55:13-56:5.

Q Understood. But in that very first instance, that initial mapping of the host name to an IP address can be done using well-known DNS processing techniques, right?

MR. FELICIELLO: Objection. Form.

A Yes. The translation of a host name to an IP address is -- on the Internet, is typically done through DNS. But, as I said, when the host name has a unique mapping to a specific IP address, it will always return the same mapping.

EX2016, 57:1-10.

Verma does not render obvious the “direct link” limitations

Verma’s “Akamai-ed” hostname-based URLs do not bypass conventional DNS resolution.

Patent Owner’s Expert

In this example, the original content URL is `http://www.foo.com/images/logo.gif`, which gets akamaized/prepended to become `http://a836.g.akamaitech.net/7/836/123/e358f5db0045e9/www.foo.com/images/logo.gif`. *Id.* As a result, instead of resolving the hostname associated with the origin server (`www.foo.com`) via DNS to obtain an IP address to the origin server node that can deliver the content (`logo.gif`), DNS would be used to resolve the hostname associated with the Akamai CDN edge server (`a836.g.akamaitech.net`) to obtain a different IP address to an edge server that can deliver a cached version of the content. *Id.*; see also EX1005.

EX2015, ¶59.



EX2017, 1.

61. Nowhere does Verma disclose an IP address or any other form of DNS-bypassing address, nor does Dr. Mowry suggest it does. This absence is not

EX2015, ¶61.

Agenda

- ▶ The '322 Patent
- ▶ “a direct link to a node within the content delivery network”
 - ▶ Independent Claims 1, 11, and 15 / All Grounds
- ▶ **The “tag” limitations**
 - ▶ **Independent Claim 1 / Grounds 3-5**
- ▶ The “application programming interface (API)” limitations (LEAP)
 - ▶ Claims 2, 9, 11, and 15

The '322 Patent, Claim 1

Claim 1

[1pre]	A content delivery method comprising:
[1a]	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;
[1b]	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; and
[1c]	delivering, by the origin system, the content resource to the client device with the modified embedded resource, 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.

Petitioner's Proposed Combination

Petitioner's theory requires Verma's content provider 22 to parse HTML pages for tags and assemble HTML pages.

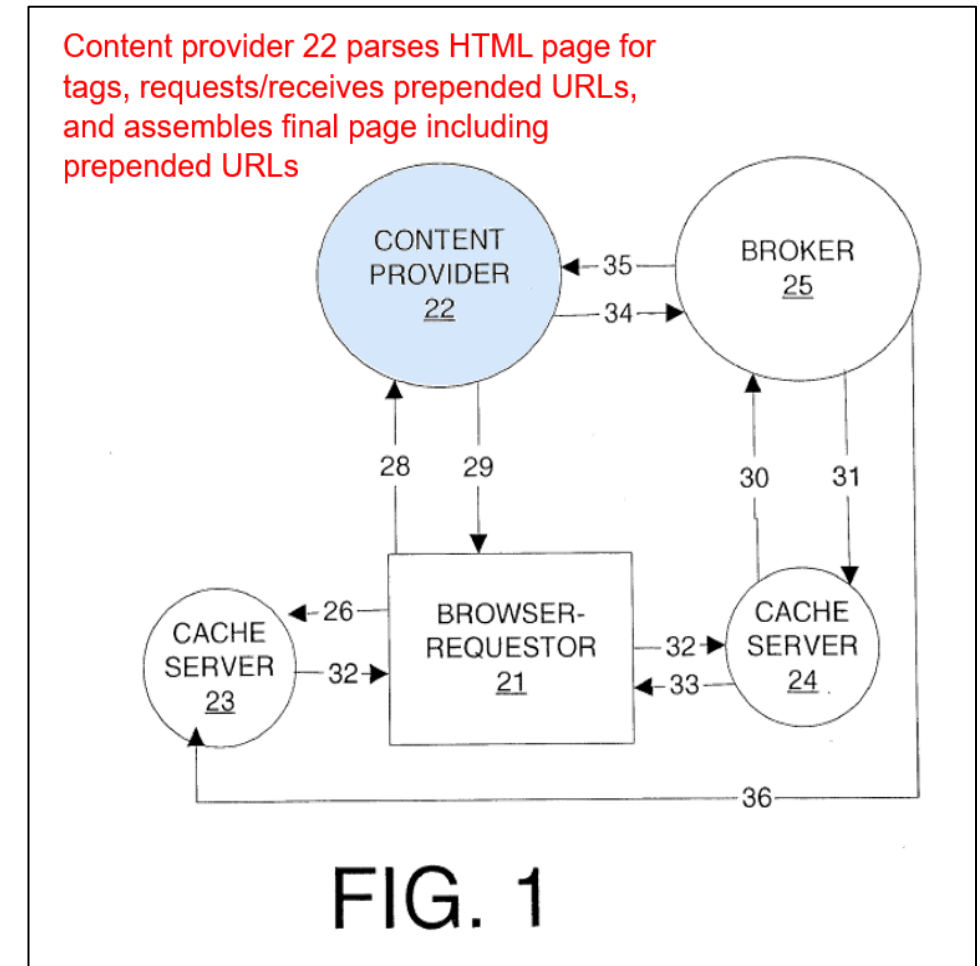
Petitioner's Combination

Patent Owner's argument distinguishing tag identification at an origin server from tag identification at an edge server (POR, 31) is also unavailing for the reasons discussed below, for it would have been obvious to parse Lewin's tags on Verma's origin system. *Infra* § III.B.2.b.

Reply, 14.

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 "prepending 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. EX-1003 ¶241; EX-1005, [0015], [0025], FIG. 1; EX-1007, [0008], [0017], [0028], [0050], FIG. 2.

Pet., 57.



PO Sur-reply, 17 (citing EX1005, FIG. 1).

Verma does not teach tag parsing or HTML page assembly

Verma does not teach HTML parsing or page assembly by content provider 22.

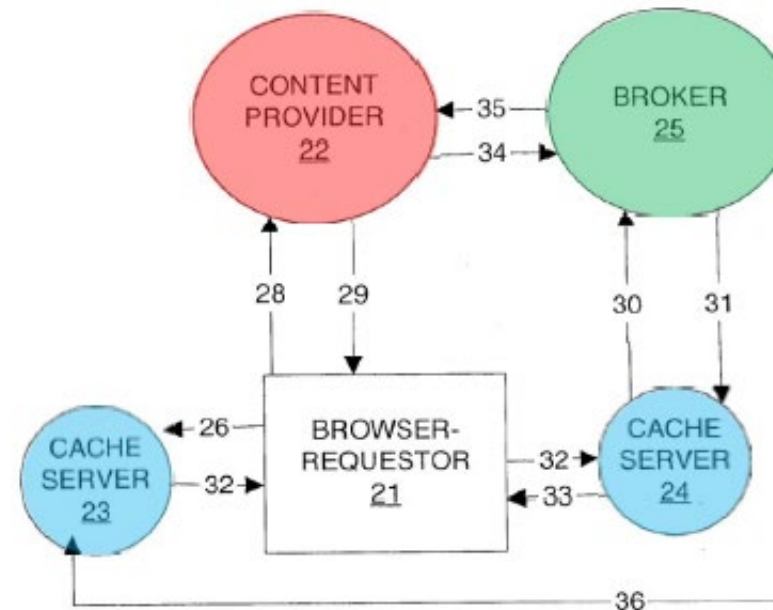
Content provider 22 merely acts as an intermediary between users and the cache servers.

Verma

[0025] FIG. 1 depicts a content delivery network in accordance with the present principles. The network includes a client that runs a browser 21. The browser 21 operates to request a content file, for example the movie entitled "Matrix Reloaded," by sending a URL 28 to a content provider 22, Movies-Inc.Com in this example. Movies-Inc.Com content provider 22 web site has a business relationship with the content delivery network to speed up the delivery of movie files, which are relatively large to its customers, such as client 21. Movies-Inc.Com content provider 22 sends a request 34 for information to a content delivery network broker 25, which manages at least one, and preferably, a set of cache servers, exemplified by cache server 23 and cache server 24. The broker 25 sends availability inquiries 31 and 36 to the cache servers 23 and 24, respectively, to find out whether they currently store the requested movie file. Various movie files are transferred from time to time from the content provider 22 to cache servers 23 and 24 via the broker 25.

[0026] The broker 25 provides content current and/or future availability information to Movies-Inc.Com content provider 22 where the content file URLs are prepended (Aka-mai-ed) for redirecting to cache servers. Additionally, infor-

EX1005, ¶26.



EX1005, FIG. 1.

EX1005, ¶25.

Patent Owner's Demonstrative Exhibit – Not Evidence

Lewin discloses page assembly at a CDN edge server

Lewin provides no basis for implementing its edge-server page assembly at Verma's content provider.

Lewin

[0003] This disclosure relates generally to content delivery over the Internet and, more specifically, to a dynamic content assembly mechanism that enable a content provider to cache, distribute and assemble individual content fragments on the edge of the Internet.

EX1007, ¶3.

dynamic page closer to requesting end users. As is well known, content delivery networks (CDNs) have the capability of caching frequently requested content closer to end users in servers located near the “edge” of the Internet. CDNs provide users with fast and reliable delivery of Web content, streaming media, and software applications across the Internet. Users requesting popular Web content may well have those requests served from a location much closer to them (e.g., a CDN content server located in a local network provider's data center), rather than from much farther away at the original Web server. By serving content requests from a server much closer electronically to the user, a quality CDN can reduce the likelihood of overloaded Web servers and Internet delays.

EX1007, ¶9.

Lewin discloses page assembly at a CDN edge server

Lewin provides no basis for implementing its edge-server page assembly at Verma's content provider.

Lewin

[0049] FIG. 9 is a flowchart illustrating the dynamic content assembly process of the present invention for an HTML page. In step (1), 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 gets sent back an IP address for a given “edge” server in the ICDN. This

EX1007, ¶49.

necessary (not shown). At step (2), the edge server parses the HTML page looking for tags that specify dynamic assembly instructions including, without limitation, URLs for HTML chunks to be incorporated in the final HTML page. The DCA tags are preferably specified according to ESI, or some other server side scripting language, as has been described generally above.

EX1007, ¶50.

Lewin discloses page assembly at a CDN edge server

Lewin provides no basis for implementing its edge-server page assembly at Verma's content provider.

Patent Owner's Expert

81. Producing a system that satisfies element [1b] using Lewin's ESI architecture would, therefore, require redesigning Lewin's techniques to account for the opposite processing direction. A POSITA would, for example, be required to design an architecture and corresponding protocol so a tag identification scheme meant to be executed server-side on the CDN edge could be efficiently and practically executed on the client origin system. This combination would also run counter to conventional edge-architecture paradigms, since the point of edge servers is to "offload[] from the origin server a significant amount of CPU processing and memory," and reduce overhead at the origin server. EX1007, ¶71.

EX2015, ¶81.

Lewin

[0071] Even serving truly non-cacheable content through the CDN is generally faster and more reliable than having customers go direct from their browser to the content provider's origin servers. The origin server preferably maintains persistent connections to a finite number of CDN edge servers, rather than trying to do this with huge numbers (perhaps millions) of individual end user browsers. A persistently-maintained connection between the origin server and the CDN speeds up requests, make the origin server generally more reliable and less variable in performance, and it offloads from the origin server a significant amount of CPU processing and memory. Performance improvements result from keeping the connection open between the edge and the origin server with data flowing through it, avoiding the overhead associated with setting up a separate connection for every browser request.

EX1007, ¶71.

An “embedded resource with a tag” is one that includes a tag

The claimed “embedded resource with a tag” is an embedded resource that includes a tag.

'322 Patent

Embodiments may also include a non-transitory computer readable media comprising an object document, such as an HTML document, comprising at least one embedded resource, the at least one embedded resource including a tag that causes the embedded resource to be analyzed to construct a modified embedded resource to replace the embedded resource, the modified embedded resource causing a browser receiving the document with the modified embedded resource to initiate communication with a content delivery network to obtain content associated with the embedded resource.

EX1001, 2:7-17.

Embodiments of the present disclosure involve a content delivery method including the operations of receiving a request for a content resource including at least one embedded resource with a tag. Upon identification of the tag, using at least one delivery parameter to generate a modified embedded resource. And, delivering the content resource with the modified embedded resource.

EX1001, 1:57-63.

The embedded resource may include a tag indicating that a resource address compute engine 30 should be contacted via a call to an API 26 (operation 220). For instance, the HTML document may include an embedded resource of the form:
http://\${CALLAPI:ADDRONLY}/path/resource.txt, or
\${CALLAPI:FULLURL,path=\$ENV{PATH},
pop=\$POP}.

EX1001, 4:32-39.

An “embedded resource with a tag” is one that includes a tag

The '322 patent discloses that tags are included within, or are part of, the corresponding embedded resource

'322 Patent

The embedded resource may include a tag indicating that a resource address compute engine 30 should be contacted via a call to an API 26 (operation 220). For instance, the HTML document may include an embedded resource of the form:

http://\${CALLAPI:ADDRONLY}/path/resource.txt, or
\${CALLAPI:F'ULLURL,path=\$ENV{PATH},
pop=\$POP}.

Tag

Embedded
Resource

EX1001, 4:32-39.

In both instances, the tag is “\$” which causes the origin server 18 to call the API 26 to retrieve the address (only) or the complete URL. Hence, the presence of the tagged resource 24A in the HTML document 20 at the origin servers causes the origin server to call the API, the result of which causes the HTML document returned to the requesting client to include a modified embedded resource 24B with an address or URL to the CDN whereby the content may be retrieved.

EX1001, 4:40-48.

An “embedded resource with a tag” is one that includes a tag

The '322 patent specification consistently uses “include” and “with” to mean the same thing

Embodiments may also include a non-transitory computer readable media comprising an object document, such as an HTML document, comprising at least one embedded resource, the at least one embedded resource including a tag that causes the embedded resource to analyzed to construct a modified embedded resource to replace the embedded resource, the modified embedded resource causing a browser receiving the document with the modified embedded resource to initiate communication with a content delivery network to obtain content associated with the embedded resource.

EX1001, 2:7-17.

embedded resource. Regardless, the HTML document 20 with the modified imbedded resource 24B is returned to the client device (operation 240).

EX1001, 5:32-34.

the complete URL. Hence, the presence of the tagged resource 24A in the HTML document 20 at the origin servers causes the origin server to call the API, the result of which causes the HTML document returned to the requesting client to include a modified embedded resource 24B with an address or URL to the CDN whereby the content may be retrieved.

EX1001, 4:42-48.

Embodiments of the present disclosure involve a content delivery method including the operations of receiving a request for a content resource including at least one embedded resource with a tag. Upon identification of the tag, using at least one delivery parameter to generate a modified embedded resource. And, delivering the content resource with the modified embedded resource.

EX1001, 1:57-63.



The interchangeable use of the two terms is akin to a definition equating the two.

Edwards Lifesciences LLC v. Cook Inc., 582 F.3d 1322, 1329 (Fed. Cir. 2009).

Lewin does not disclose an “embedded resource with a tag”

Petitioner relies on Lewin, not Verma, for the claimed “embedded resource with a tag.”

Petition

To the extent 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,” it would have been obvious to a POSITA in view of Lewin. EX-1003 ¶¶191, 193.

Lewin explains “directives [] are placed in HTML pages” and “evaluated on a server before the page is served,” which “enable[s] the server to add dynamically-generated content to an existing HTML page.” EX-1007, [0028]. Dynamic assembly instructions may be “URLs for HTML chunks to be incorporated in the final HTML page.” *Id.*, [0049]. “URLs” are embedded resources, as explained above. §§ VI.A.3, VII.A.2 (Grounds 1A/1B, 2, [15b]); EX-1007, [0028], [0049]; EX-1003 ¶197.

Tags: The “dynamic assembly instructions” (URLs (embedded resources)) are “specif[ied]” by “tags,” like a “<esi:include> tag.” EX-1007, [0028], [0030], [0049];

Lewin does not disclose an “embedded resource with a tag”

Lewin’s purported “embedded resource” (e.g., a URL) does not have a tag

Petitioner’s Mapping to Lewin

page.” *Id.*, [0049]. “URLs” are embedded resources, as explained above. §§ VI.A.3, VII.A.2 (Grounds 1A/1B, 2, [15b]); EX-1007, [0028], [0049]; EX-1003 ¶197.

Pet., 47.

Lewin discloses an embedded resource “with a tag” because the tag is next to, associated with, and identifies the embedded resource. Pet., 47-48. As the Petition

Reply, 13.

URLs. *Id.* (citing EX1007, [0028], [0030], [0049]; EX1003, ¶198). Lewin’s Figure 6 shows the placement of “tag[s]” “with” the URLs (embedded resources):

Reply, 13-14.

```
<html>
<body>
<!--personalized greeting (1)-->
Hello ${HTTP_COOKIE{"username"}}

<!--targeted ad (2)-->
<esi:choose>
  <esi:when test="$GEO{country_code})='US' ">
    <esi:include src="us_ad.html"/>
  </esi:when>
  <esi:when test="$GEO{country_code})='Canada' ">
    <esi:include src="canada_ad.html"/>
  </esi:when>
  <esi:otherwise>
    <esi:include src="generic_ad.html"/>
  </esi:otherwise>
</esi:choose>

<!--Static navigation bar (3)-->
<a href=...> <a href=...> <a href=...> <a href=...>

<!--Personalized recommendations (4)-->
<esi:include src="recommendations.html"/>

<!--Static links, copyright,etc (5)-->
<a href=...> <a href=...> <a href=...> <a href=...>
Copyright 2001, etc.
</body>
</html>
```

Purported
“embedded
resource”

EX1007, FIG. 6.

FIG. 6

Agenda

- ▶ The '322 Patent
- ▶ “a direct link to a node within the content delivery network”
 - ▶ Independent Claims 1, 11, and 15 / All Grounds
- ▶ The “tag” limitations
 - ▶ Independent Claim 1 / Grounds 3-5
- ▶ **The “application programming interface (API)” limitations (LEAP)**
 - ▶ **Claims 2, 9, 11, and 15**

The '322 Patent, Claims 11 and 15

Claim 11

[11pre]	A content delivery network comprising:
[11a]	at least one processor in communication with a non-transitory computer readable medium including computer readable instructions for:
[11b]	providing an application programming interface (API);
[11c]	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;
[11d]	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;
[11e]	delivering, by the content delivery network, the modified embedded resource to the origin system;
[11f]	receiving, at the node, a request from a client device to obtain the content; and
[11g]	providing the content to the client device.

Claim 15

[15pre]	A method comprising:
[15a]	providing, at a content delivery network, an application programming interface (API);
[15b]	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;
[15c]	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;
[15d]	delivering, by the content delivery network, the modified embedded resource to the origin system;
[15e]	receiving, at the node, a request from a client device to obtain the content; and
[15f]	providing the content to the client device.

The '322 Patent – “API”

The claimed "API" provides an independent, defined programmatic interface, regardless of whether it also includes associated compute routines and functions.

'322 Patent

delivered from the CDN. The API call may include a name of the resource, a location of the client requesting the resource and/or an IP address for the client which may be used to determine the location of the client, and/or the popularity of the resource being requested (or embedded within a resource). The API or other computing engine associated therewith generates a target or list of targets (nodes capable of serving the content from the CDN), and provides one or more locations from which the content (or embedded content) may be served. In one particular form, the API may return one or more virtual IP addresses (VIPs) for one or more clustered edge servers. The API may also return the protocol to use for requesting the content. The API may also return a list of absolute URL's from which the content may be retrieved, and/or a list of hostnames and/or IP addresses that could be used to construct the URLs at the calling application.

EX1001, 3:14-30.

Aspects of the present disclosure relate to network address resolution, and particularly involve introduction of an application programming interface into the resolution stream to embed link resolution information within an HTML document or other content returned to a requesting client.

EX1001, 1:13-18.

to the New York node. While described herein as an API call (or calls) in conjunction with an associated compute engine, other applications may be used to compute and return the modified embedded resource. In some instances, only an API is referenced but it should be recognized that such reference includes not only the interface but also the associated compute routine, function, and the like for processing inputs to the API and generating a modified embedded resource. In any event, a modified embedded resource is generated and substituted for the resource, where the modified resource identified a node able to serve the content (operation 230).

EX1001, 5:4-15.

The '322 Patent – “API”

Patent Owner's Expert

106. To add more clarity, 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. Industry-standard technical literature confirms that these defining characteristics include: (1) the interface is accessible to an external caller (here, the origin server) via an “endpoint”—“[s]pecific URLs that define where to send data and requests”; (2) it accepts structured inputs through defined “parameters”—“[s]pecific details needed for the request”; and (3) it returns a defined output in “response”—“[t]he format of the data sent back by the application.” EX2019, 2–3. These characteristics are

defined through an “API specification [that] provides a structured description of what the API will do and exactly how to interact with it.” *Id.*, 3. This is precisely

EX2015, ¶106.

An API is a middleman that allows two software programs to talk to each other and defines how they request and receive data or functionality.

EX2019, 2.

An API defines how applications interact by providing details including:

Endpoints. Specific URLs that define where to send data and requests.

Methods. Instructions such as GET to retrieve data, POST to send data, PUT to update data, and DELETE to remove data.

Parameters. Specific details needed for the request, such as location for weather data or login credentials for social media.

Responses. The format of the data sent back by the application, such as JSON or XML.

EX2019, 3.

Petitioner does not offer an alternative definition for the claimed API

Petitioner Reply

Patent Owner instead focuses on this language's placement following "[i]n an alternative implementation" to contend it is meant to signal an embodiment in which "the CDN handles address resolution internally" instead of by "call[ing] an external API exposed by the CDN." POR, 42-43. In Patent Owner's view, the claims cover the latter. *Id.* But when properly read, the paragraph explains other implementations exist, some implementations "may also involve an API," and "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. That an alternative might not use "a customer origin server or other devices outside the CDN" (*id.*) does not detract from express disclosure that an API, as contemplated by the specification and claims, can comprise the engine.

Pet. Reply, 8.

Finally, Patent Owner contends that "API" is rendered superfluous unless it is discrete from the computing engine. POR, 43-44. Not so. Claims 11 and 15 recite an API that receives information for the CDN, but the claims flexibly leave open where generating the modified embedded resource may occur (i.e., at the API or elsewhere in the CDN). EX1001, 10:59-11:9, 12:5-20. This comports with the specification's definition that an API may or may not include the compute-engine functionality. *Id.*, 7:16-25. "API" is not superfluous if the Board's construction is adopted.

Pet. Reply, 9.

CDN." *Id.*, 5:65-6:23. So too may the API provide outputs in different forms: "API returns an absolute URL ..., and/or a list of hostnames and/or IP addresses." *Id.*, 6:50-53. Requiring a specific interface having a "specified input-output contract," as Patent Owner contends, introduces concepts found nowhere in the claim language or specification. POR, 51.

Pet. Reply, 10.

The claimed API requires a defined interface

'322 Patent

dictatorily 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

EX1001, 7:20-23.

to serve content to a client. Aspects of the present disclosure 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

EX1001, 3:9-12.

to the New York node. While described herein as an API call (or calls) in conjunction with an associated compute engine, other applications may be used to compute and return the modified embedded resource. In some instances, only an API is referenced but it should be recognized that such reference includes not only the interface but also the associated compute routine, function, and the like for processing inputs to the API and generating a modified embedded resource. In any event, a modified embedded resource is generated and substituted for the resource, where the modi-

EX1001, 5:4-13.

Petitioner Reply

Finally, Patent Owner contends that “API” is rendered superfluous unless it is discrete from the computing engine. POR, 43-44. Not so. Claims 11 and 15 recite an API that receives information for the CDN, but the claims flexibly leave open where generating the modified embedded resource may occur (i.e., at the API or elsewhere in the CDN). EX1001, 10:59-11:9, 12:5-20. This comports with the specification’s definition that an API may or may not include the compute-engine functionality. *Id.*, 7:16-25. “API” is not superfluous if the Board’s construction is adopted.

Pet. Reply, 9.

The API's defined interface is also separate functionality from the compute engine's

'322 Patent

[11c]	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;
[15b]	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;

to the New York node. While described herein as an API call (or calls) in conjunction with an associated compute engine, other applications may be used to compute and return the modified embedded resource. In some instances, only an API is referenced but it should be recognized that such reference includes not only the interface but also the associated compute routine, function, and the like for processing inputs to the API and generating a modified embedded resource. In any event, a modified embedded resource is generated and substituted for the resource, where the modified resource identified a node able to serve the content (operation 230).

EX1001, 5:4-15.

Petitioner Reply

Patent Owner instead focuses on this language's placement following "[i]n an alternative implementation" to contend it is meant to signal an embodiment in which "the CDN handles address resolution internally" instead of by "call[ing] an external API exposed by the CDN." POR, 42-43. In Patent Owner's view, the claims cover the latter. *Id.* But when properly read, the paragraph explains other implementations exist, some implementations "may also involve an API," and "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. That an alternative might not use "a customer origin server or other devices outside the CDN" (*id.*) does not detract from express disclosure that an API, as contemplated by the specification and claims, can comprise the engine.

Pet. Reply, 8.

There would be no reason to recite a separate API if the API and the engine were the same component performing the same functionality.

The claimed API requires a defined input-output relationship

Petitioner Reply

CDN.” *Id.*, 5:65-6:23. So too may the API provide outputs in different forms: “API returns an absolute URL ..., and/or a list of hostnames and/or IP addresses.” *Id.*, 6:50-53. Requiring a specific interface having a “specified input-output contract,” as Patent Owner contends, introduces concepts found nowhere in the claim language or specification. POR, 51.

Pet. Reply, 10.

'322 Patent

In any event, the API 26 provides the interface to a computing engine 30 that uses the information, passed to the engine through the API, to determine one or more nodes 32 in the CDN to serve the content upon receiving a request from the origin server (or from some other request source) related to a tagged resource, or resource otherwise identified for modification (operation 310). The computing engine

EX1001, 6:24-30.

An API is a middleman that allows two software programs to talk to each other and defines how they request and receive data or functionality.

EX2019, 2.

An API defines how applications interact by providing details including:

Endpoints. Specific URLs that define where to send data and requests.

Methods. Instructions such as GET to retrieve data, POST to send data, PUT to update data, and DELETE to remove data.

Parameters. Specific details needed for the request, such as location for weather data or login credentials for social media.

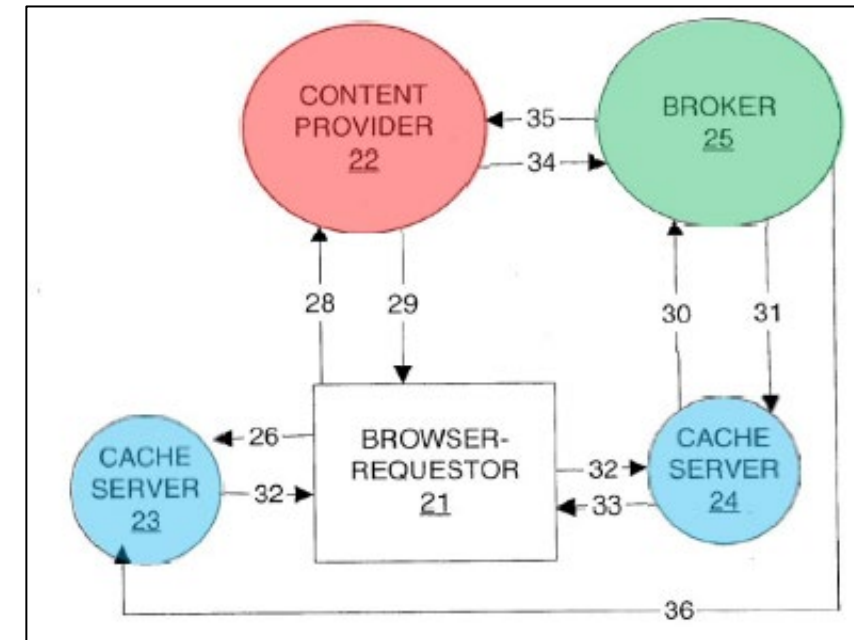
Responses. The format of the data sent back by the application, such as JSON or XML.

EX2019, 3.

Verma does not teach an API

Verma

[0025] FIG. 1 depicts a content delivery network in accordance with the present principles. The network includes a client that runs a browser 21. The browser 21 operates to request a content file, for example the movie entitled “Matrix Reloaded,” by sending a URL 28 to a content provider 22, Movies-Inc.Com in this example. Movies-Inc.Com content provider 22 web site has a business relationship with the content delivery network to speed up the delivery of movie files, which are relatively large to its customers, such as client 21. Movies-Inc.Com content provider 22 sends a request 34 for information to a content delivery network broker 25, which manages at least one, and preferably, a set of cache servers, exemplified by cache server 23 and cache server 24. The broker 25 sends availability inquiries 31 and 36 to the cache servers 23 and 24, respectively, to find out whether they currently store the requested movie file. Various movie files are transferred from time to time from the content provider 22 to cache servers 23 and 24 via the broker 25.



EX1005, FIG. 1.

EX1005, ¶25.

Verma is silent on the structure of its request, and Petitioner does not point to any API exposed by Verma’s broker.

Verma's “broker” is not the claimed API

Petition

Application Programming Interface: Verma’s broker is an API, application programming interface, because it “communicate[s] with the content server(s)” and

provides an interface between content server(s) and CDN. EX-1005, [0015], [0022], [0025]-[0026]; EX-1003 ¶87. For example, “content provider 22 sends a request 34 for information to a content delivery network broker 25,” and “broker 25 provides” information to “content provider 22.” EX-1005, [0025]-[0026]; EX-1003 ¶87. 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, which provides that “[w]hen the origin server receives the request,” it “calls an API” to “communicate with the compute engine” in the CDN. EX-1001, 4:49-52, 7:16-25 (“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”; “the API may include the functionality of the engine”); EX-1003 ¶87.

Pet., 11-12.

Petitioner Reply

Second, Verma discloses defined inputs and outputs. For example, Verma’s content provider “request[s]” the broker “for a content file delivery to a location close to the client” and provides delay tolerance. EX1005, [0015], [0025]-[0026]. “[L]ocation” and delay tolerance are defined parameters that Verma’s content provider provides to the broker to obtain a modified embedded resource. *Id.* Further, Verma’s broker returns a prepended URL to the content provider, and that prepended URL is a defined output. Pet., 25-26; EX1005, [0012], [0022], [0026], cl. 18;

Pet. Reply, 27.

Petitioner's Reply fails to cure the Petition's deficiencies

A “business relationship” and “negotiation” are not a defined API.

Petitioner Reply

This fails to capture Verma's full disclosure. Verma states “content server 22 will negotiate with the content delivery network (CDN) through the broker 25,” and the content provider “has a business relationship with the [CDN],” confirming Verma's broker is a dedicated interface for the content provider (origin system) to contact the CDN. EX1005, [0038]; Pet., 13-14 (quoting EX1005, [0025]; EX1003, ¶92). Like the '322 patent, Verma's broker provides “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.” EX1001, 7:16-20; Pet., 11-12; EX1003, ¶87.⁶

Pet. Reply, 26.

'322 Patent

dictably 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

EX1001, 7:20-23.

to serve content to a client. Aspects of the present disclosure 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

EX1001, 3:9-12.

At most, Verma involves communication and passing information between content provider and broker.

Verma does not disclose a defined programmatic interface

Patent Owner's Expert

108. In other words, merely receiving and forwarding a request through an internal CDN pipeline—without a defined interface contract, defined parameters, or a defined API response format—is not an API call. As a POSITA would have recognized, 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. The distinction is whether the external caller invokes a specific, defined interface with a specified parameter-response contract, or whether it simply sends a message into a system's internal pipeline. 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, including what interface definition (if any) governs the exchange. EX2016, 36:15-37:12. Verma discloses no defined programmatic interface, and Dr. Mowry does not point to one.

EX2015, ¶108.

And with a *dynamically* generated response, isn't a simple http call little different performance-wise to an API?

True, from a pure performance perspective, a simple HTTP call that dynamically generates a response and a more structured API call that does the same thing can have similar performance. The actual performance in either case largely depends on how efficiently the server handles the request, generates the response, and the nature of the operations involved (database access, computations, etc.).

However, as mentioned earlier, APIs are not only about serving data. They provide a contract, documentation, standardization, versioning, and many other features that a standalone HTTP call may not provide. These additional features may add some overhead, but they also provide considerable benefits in terms of maintainability, scalability, and ease of use.

EX2020, 5.