IN THE UNITED STATES DISTRICT COURT

FOR THE DISTRICT OF DELAWARE

ORACLE CORPORATION and ORACLE U.S.A. INC. Plaintiffs/Counterdefendants, v. EPICREALM LICENSING, LP, Defendant/Counterclaimant.	)	C.A. No. 06-414 (SLR)
AND RELATED COUNTERCLAIMS	)	

DECLARATION OF DR. MICHAEL IAN SHAMOS IN SUPPORT OF ORACLE'S MOTION FOR SUMMARY JUDGMENT OF INVALIDITY OF THE ASSERTED CLAIMS (CORRECTED)

MORRIS, NICHOLS, ARSHT & TUNNELL LLP
Mary B. Graham (#2256)
James W. Parrett, Jr. (#4292)
1201 North Market Street
P.O. Box 1347
Wilmington, DE 19899
(302) 658-9200

Attorneys for Oracle Corporation and Oracle U.S.A. Inc.

OF COUNSEL:

James G. Gilliland
Theodore T. Herhold
Joseph A. Greco
Robert J. Artuz
Eric M. Hutchins
TOWNSEND AND TOWNSEND AND CREW LLP
379 Lytton Avenue
Palo Alto, CA 94301
(650) 326-2400

Dorian Daley
Peggy E. Bruggman
Matthew M. Sarboraria
ORACLE CORPORATION
ORACLE U.S.A., INC.
500 Oracle Parkway
Redwood Shores, CA 94065

Originally filed (part of D.I. 217): July 31, 2008
Corrected (as to paragraph numbers only): August 20, 2008

I, Dr. Michael Ian Shamos, declare as follows:

I. INTRODUCTION

1. I have been engaged by counsel for Plaintiffs Oracle Corporation and Oracle U.S.A. (collectively, "Oracle") in this case to provide expert opinions on the validity of claims 1-11 of U.S. Patent No. 5,894,554 (the "'554 Patent") and claims 2 and 16 of U.S. Patent No. 6,415,335 (the "'335 Patent"), collectively the "Asserted Claims" of the "epicRealm patents," based on the claim constructions propounded by Oracle and epicRealm. If called and sworn as a witness, I could and would testify competently to the opinions stated in this Declaration.

II. MY BACKGROUND

2. I hold the title of Distinguished Career Professor in the School of Computer Science at Carnegie Mellon University in Pittsburgh, Pennsylvania. I was a founder and Co-Director of the Institute for eCommerce at Carnegie Mellon and I now direct a graduate degree program in eBusiness Technologies. Attached as A31 to the Appendix[1] of Exhibits accompanying Oracle's motions for summary judgment is a true and correct copy of my Curriculum Vitae.

3. I currently teach graduate courses at Carnegie Mellon in Electronic Commerce, including eCommerce Technology, Electronic Payment Systems, Electronic Voting and eCommerce Law and Regulation and have done so since 1999. In Fall 2007 I taught a new course entitled Law of Computer Technology, which I will also be teaching this year.

4. From 1979-1987, I was the founder and president of two computer software development companies in Pittsburgh, Pennsylvania, Unilogic, Ltd. and Lexeme Corporation.

[1] References to exhibits of the Appendix will be referred to herein as "A_".

5. I am an attorney admitted to practice in Pennsylvania and have been admitted to the Bar of the U.S. Patent and Trademark Office since 1981. I have not been asked to offer any opinions on patent law in this action.

6. I have previously testified in a number of cases concerning computer software, intellectual property, electronic payment, electronic voting and electronic commerce matters. My C.V. in A31 contains a list of cases in which I have testified in the last ten years.

7. I have also been engaged by defendants in three cases in the Eastern District of Texas ("Texas Actions"). The Texas Actions assert that the various defendants are infringing the same Asserted Claims of the epicRealm patents. I have submitted an expert report on invalidity in those cases that is substantially similar to the one I submitted in this case.

8. The individuals named as inventors in the epicRealm patents are referred to collectively herein as the "Applicants."

9. In this Declaration, where I have cited a reference as prior art, either the reference predates the filing date of the epicRealm patents or I have been informed by counsel for Oracle that Oracle will be able to prove at trial that the reference is prior art as to the epicRealm patents.

III. LEGAL PRINCIPLES

10. I have not been asked to offer an opinion on the law; however, as an expert assisting the Court in determining claim validity, I understand that I am obliged to follow existing law. I have therefore been asked to apply the following legal principles to my analysis of allegations of invalidity in light of the prior art.

11. For a claim to be anticipated, every limitation of the claimed invention must be found in a single prior art reference, either expressly or inherently, arranged as in the claim.

12. When a claim covers several alternative structures or compositions of elements, either generically or as alternatives, the claim is deemed anticipated if any of the structures or

compositions within the scope of the claim is disclosed or practiced in the prior art.

13. For a claim element to be inherently present in a prior art reference, the element must be "necessarily present" in the disclosed apparatus, system or method, not merely probably or possibly present.

14. A claim is invalid for obviousness if differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains.

15. In determining whether a claimed invention is obvious, one should consider the scope and content of the prior art, the level of ordinary skill in the relevant art, the differences between the claimed invention and the prior art, and whether the claimed invention would have been obvious to one of ordinary skill in the art in light of those differences.

16. I understand that certain objective factors, sometimes known as "secondary considerations" may also be taken into account in determining whether a claimed invention would have been obvious. One of these factors is contemporaneous development of the claimed invention by others. I have reviewed epicRealm's Response to Interrogatory No. 12 regarding "secondary considerations" and do not believe that such alleged considerations weigh in favor of nonobviousness, especially in view of the numerous prior art references discussed herein and in my expert report which anticipate the Asserted Claims and the numerous individuals and companies, including Oracle, discussed or referenced herein or in my report which developed the claimed inventions of the epicRealm patents prior to their filing dates.

17. The person of ordinary skill is a hypothetical person who is presumed to be aware of all of the pertinent art. The person of ordinary skill is not an automaton, and may be able to fit together the teachings of multiple patents employing ordinary creativity and the common sense

that familiar items may have obvious uses beyond their primary purposes. A patent which merely claims predictable uses of old elements according to their established functions to achieve predicable results should be found invalid as obvious.

18. In establishing obviousness, one must avoid the temptation to read into the prior art the teachings of the invention in issue and guard against slipping into the use of hindsight.

19. An invention is obvious if a designer of ordinary skill in the art, facing the wide range of needs created by developments in the field, would have seen an obvious benefit to the solutions tried by the applicant.

IV. THE EPICREALM PATENTS

20. The specifications of the epicRealm patents are substantially identical and therefore it makes sense to treat them collectively when describing their subject matter.

21. In the mid-1990s, most Web pages consisted of relatively fixed content that was not modified very frequently, e.g., less often than once per day. Such pages were referred to as "static." It made sense to store static pages on disk drives or in random-access memory so they could be delivered rapidly to Web clients that requested them. A typical example of a static page is the home page of a company giving various details such as the company's address and telephone number, along with link to other static pages that provide information about the company's products.

22. Static pages, of course, are not suitable for information which changes frequently, such as the price of a stock, or for information that is retrieved or computed on a custom basis at the moment it is requested. Such pages are called "dynamic pages" because they are created dynamically on request. A good example of a dynamic page is an individual user's shopping cart at a retail website. Obviously the shopping cart cannot be prestored because it is created dynamically in response to the shopping activity of the user.

23. Because dynamic pages are created when they are requested, more computer resource is needed to produce and deliver a dynamic page than for a static page. It may in fact take thousands of times longer to generate a dynamic page. While Web servers can be extremely efficient at delivering static pages, their performance becomes unacceptably slow if they are also burdened with the computational load needed for dynamic pages. In order to maintain acceptable response times, it was known to send dynamic page requests to some server other than the Web server, restricting the Web server to handling only static requests.

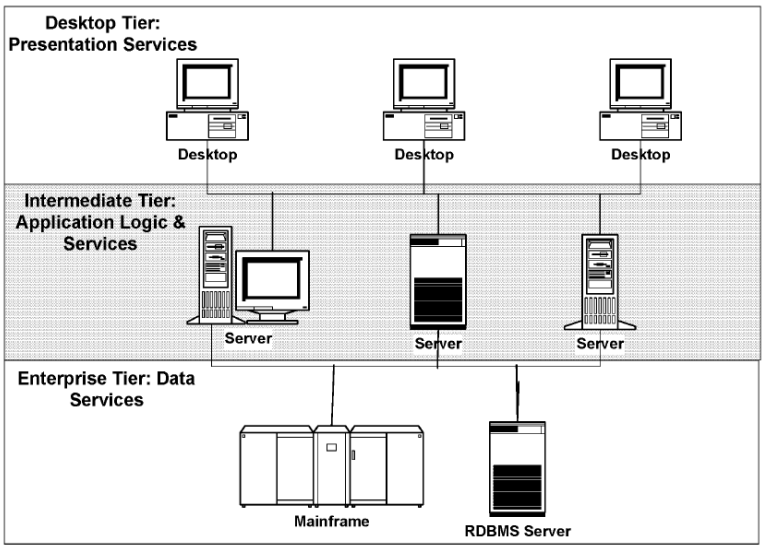
24. In this Declaration, the phrase "it was known" means throughout that it was known prior to the inventions claimed in the epicRealm patents.

25. The epicRealm patents deal with methods for handling requests for dynamic pages by passing them off to one or more separate servers known as "page servers." To do this, the epicRealm patents make use of a well-known system architecture known as the "three-tier architecture." In January 1995, Wayne W. Eckerson published a paper in the journal Open Information Systems entitled, "Three Tier Client/Server Architecture: Achieving Scalability, Performance, and Efficiency in Client Server Applications" ("Eckerson"). See Appendix, A11. A figure from that paper is reproduced below. Tier 1 (called the "Desktop Tier" in the diagram) comprises the Web client, i.e., the computer on which a Web browser is running. Tier 2 (called the "Intermediate Tier" in the diagram) comprises the Web server. Tier 3 (the "Enterprise Tier") is shown as having a mainframe computer and a relational database management (RDMBS) server.

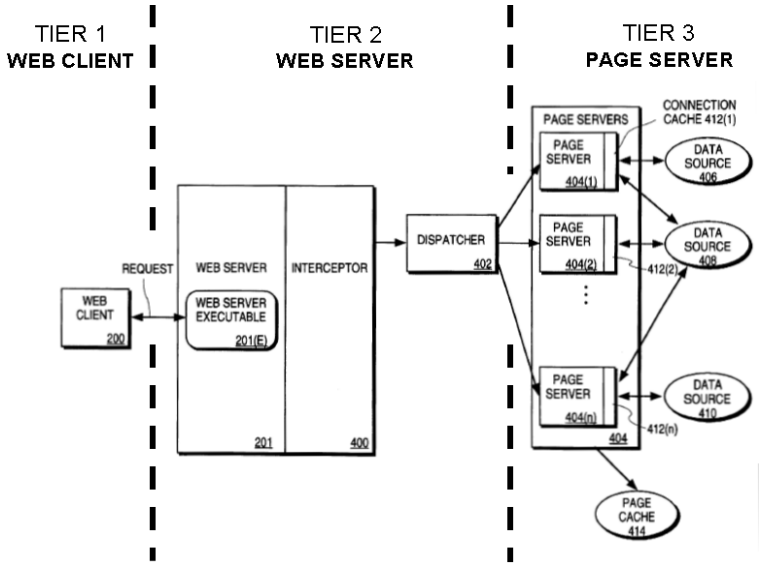
26. Static pages can be served through the Tier 2 servers (Web servers). Resource-intensive work, such as the generation of dynamic web pages, including retrieving dynamic data from databases and adding HTML markup, is performed by computers at Tier 3. The nature of the three-tier architecture is that the Tier 3 servers perform the heavy work of handling dynamic

page requests, allowing the Tier 2 (Web) servers to continue rapid delivery of static pages.

Three-Tier Deployment Architecture



27. That the epicRealm patents employ a standard three-tier architecture is apparent from Fig. 4 of the '554 Patent, reproduced below. The dotted vertical lines and the "Tier" headings have been added; otherwise, the drawing is unaltered.



28. While the diagram from the Eckerson reference shows the tiers drawn vertically, as a wedding cake, the figure from the epicRealm patents shows the tiers horizontally. However, there is no essential difference between the architectures.

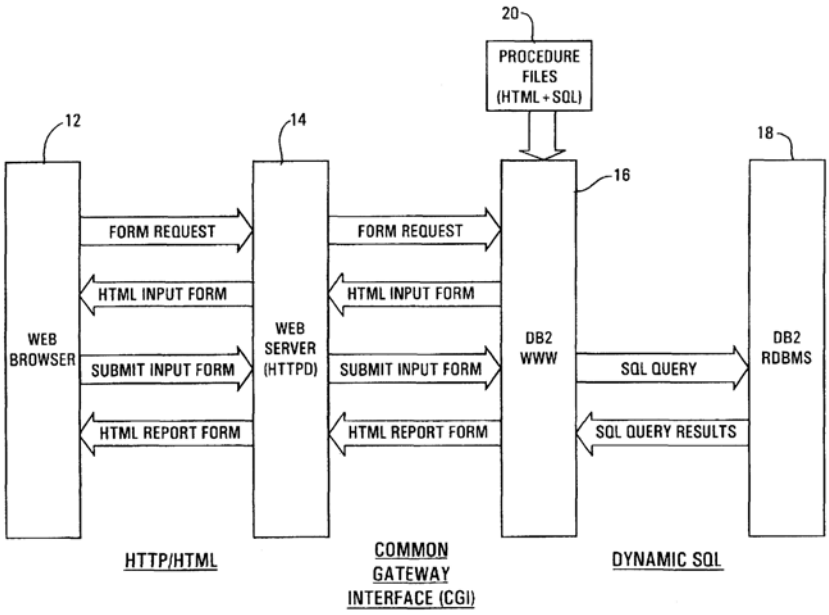
29. The figure from the epicRealm patents shows two boxes that are not expressly present in the Eckerson diagram, an "interceptor" and a "dispatcher." The function of the interceptor is to keep the Web server from spending time on dynamic page requests. Per my understanding of epicRealm's infringement theory for the "interceptor" under both parties' proposed constructions for that term, it need only discriminate between requests that are to be handled by the Web server itself (static pages) and those that must be handed off to page servers (dynamic pages). Note that exactly the same function is performed in the three-tier architecture of Eckerson, even though no box specifically labeled "interceptor" is shown. This is because a decision must be made whether a request made by the Web client is to be handed by the Intermediate Tier (2) or the Enterprise Tier (3).

30. Because Tier 3 in both Eckerson and the epicRealm patents comprises more than one server, some decision must be made which server should receive the request. Many methods of making such a decision are possible, including making a random choice. Eckerson, however, teaches making an intelligent choice based on a principle called "load balancing." A11 at ORCL01043758.

31. The intelligent choice of which Tier 3 server should receive the request is called "dispatching" in the epicRealm patents. The function of a dispatcher is performed in Eckerson by the code which implements the quoted failover and load-balancing logic in Tier 2. This is true even though there is no box expressly labeled "dispatcher" in the Eckerson diagram.

32. The "data sources" accessed by Tier 3 of the epicRealm patents can sometimes be regarded as a fourth tier, resulting in a four-tier architecture. Four-tier architectures predated the epicRealm patents. Such an architecture for handling relational database queries over the Internet is disclosed in Ngyuen et al. U.S. Patent 5,737,592, "Accessing a relational database over the Internet using macro language files," having an application date of June 19, 1995. Fig.

2 of Nguyen et al., shown below, clearly indicates the four tiers.



33. Nguyen et al. discloses a fundamental structure of the epicRealm patents, namely the passing off of dynamic database queries to a separate server, interrogating the database, and substituting the results of the query into an HTML template, the result of which is returned to the user's web browser for display. The "page servers" and "data sources" of the epicRealm patents correspond to the "DB2 WWW" and "DB2 RDBMS" tiers in the Nguyen et al. drawing.

V. THE PRIOR ART

34. To determine what prior art is relevant, we must define the art. Both the '554 and the '335 patents define the art as follows: "The present invention relates to the field of Internet technology. Specifically, the present invention relates to the creation and management of custom World Wide Web sites." '554 patent, col. 1:9-11 (Appendix, A1). The problem to be solved is further described later in the epicRealm patents: "Current Web server architecture also does not allow the Web server to efficiently manage the Web page and process Web client requests. Managing these hundreds of Web pages in a coherent manner and processing all requests for access to the Web pages is thus a difficult task. Existing development tools are limited in their

capabilities to facilitate dynamic Web page generation, and do not address the issue of managing Web requests or Web sites." '554 patent, col. 2:4-12.

35. The level of skill of one skilled in the art to which the epicRealm patents pertain would require familiarity with Internet programming, Web servers, methods of generating dynamic Web pages and familiarity with relevant Internet standards and protocols. No particular college degree would be required, nor would a person need to be a computer professional. Understanding the relevant technology presupposes programming skills but no advanced training.

36. The epicRealm patents address a problem that was well known at the time the applications were filed and for which many solutions were known, including the very matter claimed in the epicRealm patents.

37. A web page is "requested" by a client computer and transmitted over the Internet by a server computer, usually just called a "server," or more precisely for purposes of this Declaration, a "web server." The job of the web server is to receive requests for web pages and service those requests as rapidly as possible by transmitting the requested pages. Static pages can be accessed and delivered quickly by a web server. Any delay results in a slowdown, which degrades the performance of the website. Clients who are requesting pages will experience delays if the server is unable to handle the requested volume of requests. Therefore, it was well known in the art to "tune" servers to make them as fast as possible. One manner of tuning was to eliminate all application software on the web server that was not absolutely necessary for delivery of web pages.

38. Another problem arose from the practical fact that every computer, no matter how fast, can only process a certain number of requests per second. If more are anticipated, a single machine will not be sufficient to provide adequate response. It was therefore necessary to

employ more than one web server at a web site to handle the expected volume of traffic. Eventually, large collections of web servers, called "server farms," were deployed. The existence of more than one server led to the problem of determining which server a specific request should be set to, and how to get it there. One possible protocol was to keep track of the "load" on each server in the farm, that is, monitoring how much spare response capacity was available on each server at short intervals. A newly-arrived request would then be sent, or "dispatched," to the server that was least busy. This was one known method of "load-balancing," or trying to keep all the servers operating at approximately the same load. Load balancing is an old art that developed long before the World Wide Web. It was initially used in multiprocessor systems (systems incorporating more than one computer) to determine what processing ought to be done on each machine to make effective use of multiprocessor power.

39. Soon after the World Wide Web was developed, it became apparent that static pages were not adequate to satisfy users' information needs. A banking customer, for example, wants to see his own account data, not some general page of bank information. It is infeasible to store static pages containing all possible pages all bank customers might want to see and keep them constantly updated. The solution was the development of the "dynamic page," a page that is not yet in existence at the time it is requested, but is created to order in real-time as needed. Dynamic pages were known long before the epicRealm patents were applied for.

40. A problem with dynamic pages is that they take time to create. The request must be analyzed to determine what kind of page is needed. Generally, data has to be retrieved from a database to populate fields on the page, which must then be formatted properly before it can be transmitted to the client. This process is vastly slower than delivering a static page out of random-access memory. The need to deliver dynamic pages caused huge degradation in the performance of web servers. The solution, known long before the epicRealm patents, was to

analyze a request on the web server only long enough to determine whether it asks for a static or a dynamic page. If a static page is requested, the request can be fulfilled by the web server itself, by transmitting the pre-stored page, either from disk or from random-access memory. If a dynamic page is requested, the request is passed on to a different computer (a "page server") whose job is to retrieve data and construct the page. When the page server completes construction of the page, it passes it back to the web server, which then delivers it to the client.

41. At the time the application for the epicRealm patents was filed, on April 23, 1996, the World Wide Web was already in public use for electronic commerce transactions and delivery of dynamic content. The Internet boom had begun, and Amazon.com had already been incorporated for two years. Web servers, server farms, load balancing and page servers were already well known and the problem purported to be solved by applicants had already been solved using the very methods claimed in the epicRealm patents. While the epicRealm patents introduce non-standard terminology for components of a four-tier architecture, such as "interceptor," "dispatcher," and "page server," the differences are linguistic only and the subject matter of the epicRealm patents is a straightforward use of the well-known multi-tier architecture. The inventions claimed in the epicRealm patents are not novel, regardless of the language in which it they may be expressed.

VI. ALL ASSERTED CLAIMS ARE ANTICIPATED

42. All of the Asserted Claims of the epicRealm patents are anticipated by the prior art. The number of references that establish anticipation is large, in some cases as many as 30, and the number of combinations that establish obviousness is in the thousands. Many of these references on their face demonstrate that epicRealm's claimed inventions were known to and

were being widely used by the public prior to epicRealm's alleged invention date of August 1995.[2] Many of these references are discussed in my expert report of invalidity. For purposes of brevity, however, I have limited the references discussed in this Declaration to a few exemplary references including:

- Oracle WebServer 1.0 (including the Oracle7 Server) ("OWS 1.0");
- Oracle WebServer 2.0 (including the Oracle7 Server) ("OWS 2.0");
- Popp et al., U.S. Patent No. 6,249,291 (filed September 22, 1995) ("Popp");
- C. Lagoze et al., *Dienst: Implementation and Reference Manual*, Cornell University Technical Report TR95-1514 (May 5, 1995) ("Dienst"); and
- M. Garland et al., *Implementing Distributed Server Groups for the World Wide Web*, School of Computer Science, Carnegie Mellon University, Technical Report CMU-CS-95-114 (January 25, 1995) ("Garland").

The fact that in this Declaration I have only relied on a few invalidating references of the many invalidating references upon which I rely in my report does not mean that I believe these selected references to be particularly better than any other prior art reference. The invalidating references I have chosen to use are discussed in more detail below.

A. All the Asserted Claims Are Anticipated by Oracle WebServer 1.0

43. As demonstrated below, the Oracle WebServer 1.0 system (including the Oracle7

[2] Although epicRealm asserts that it first conceived of its claimed inventions in August 1995, I have reviewed epicRealm's proffered evidence to support this assertion, including inventor testimony and supporting documents (EPIC000001-173), and do not believe such evidence supports an invention date earlier than the filing date of April 23, 1996. For example, none of the documents on which epicRealm relies to support an early conception or reduction to practice date discloses the claimed step of the page server "releasing" the Web server, which all the Asserted Claims require.

server) ("OWS 1.0") anticipates all of the Asserted Claims of the epicRealm patents.

Claim 1 of the '554 patent

A computer-implemented method for managing a dynamic Web page generation request to a Web server, said computer-implemented method comprising the steps of:

44. As stated in the Oracle WebServer ("OWS") 1.0 User's Guide, OWS 1.0 managed dynamic Web page generation requests to a Web server (the "HTTP server" or "Web Listener" of OWS 1.0): "The Oracle WebServer is an HTTP server with a tightly integrated Oracle7 server that enables the creation of dynamic HTML documents from data stored in an Oracle database." A40 at ORCL000021.

routing said request from said Web server to a page server, said page server receiving said request and releasing said Web server to process other requests,

45. OWS 1.0 discloses routing a dynamic Web page generation request[3] from a Web server (the Web Listener)[4] to a page server (an Oracle7 Server). First, as shown in the diagram on page *Id.* at ORCL000022, the dynamic page generation request is received by the Web Listener in the form of a URL and is routed to the Oracle7 Server. Naturally, the Oracle7 Server receives the request. "The Oracle7 Server provides the storage for all dynamic data in relational tables, and all the program logic used to create dynamic HTML pages." *Id.* at ORCL000021. Accordingly, the Oracle7 Server is a "page server" of the epicRealm patents under both parties'

---

[3] This "request" meets both parties' proposed constructions for that term, namely, "a message that asks for a Web page specified by a URL" as proposed by Oracle and, accordingly, "a message that asks for a Web page" as proposed by epicRealm.

[4] The OWS 1.0 Web Listener meets both parties' proposed constructions for "Web server," namely, an "HTTP-compliant server software, or a machine running such software, that receives Web page requests and returns Web pages in response to the requests" as proposed by Oracle and, accordingly, "software, or a machine having software, that receives Web page requests and returns Web pages in response to the requests" as proposed by epicRealm.

proposed constructions for that term.[5] Particularly, the Oracle7 Server is a "page server" under epicRealm's proposed constructions because, as discussed above, it includes software for generating "dynamic HTML pages." *Id.* Similarly, the Oracle7 Server is a "page server" under Oracle's proposed construction for that term because ORCL000021 discloses the server running on a different machine (i.e., different processor) from that of the Web Server (the Web Listener).

46. Additionally, OWS 1.0 discloses "releasing said Web Server to process other requests" under both parties' proposed constructions for that claim term.[6] Particularly, as stated in the OWS 1.0 User's Guide: "For maximum performance the Web Listener is designed to run as an asynchronous engine … providing high performance under a heavy load." *Id.* at ORCL000055. Here "asynchronous" means that it functions independently of other tasks, which are being performed simultaneously. For the Web Listener (web server) to run asynchronously, it must be released to respond to other requests after passing off a request to a page server. This meets epicRealm's implicit releasing theory under epicRealm's proposed construction for "releasing."[7] See A35 (*epicRealm's Response to Oracle's Fourth Set of Interrogatories Nos. 22-25 dated March 10, 2008*) at pp. 5-6 (explaining epicRealm's infringement theories with respect to the "releasing" claim limitation). Additionally, OWS 1.0 passes requests using HTTP, and

---

[5] The parties' proposed constructions for "page server" are listed in CHART 1 attached to Oracle's Motion for Summary Judgment of Invalidity.

[6] The parties' proposed constructions for "page server . . . releasing said Web server . . ." are listed in CHART 1 attached to Oracle's Motion for Summary Judgment of Invalidity.

[7] I understand that epicRealm has asserted at least two infringement theories with respect to the "releasing" claim limitation in both the Texas Actions and in this action: an implicit releasing theory and an explicit releasing theory. First, under epicRealm's proposed "releasing" construction (i.e., "freeing"), it has asserted an implicit releasing theory. Under this theory, the "freeing" of the accused Web server by the page server can occur simply by virtue of the fact that the page server receives the request and, as a result, implicitly releases the Web server to process other requests. I disagree with this construction of "releasing," because according to the claims of the patent, the page server must release the Web server via an affirmative act.

discloses SQL*Net running over a TCP/IP connection, so TCP releasing is performed under epicRealm's explicit releasing theory.[8] Furthermore, the Oracle7 Server binds to a TCP port and therefore the releasing limitation is met under epicRealm's SBL theory of releasing. A40 at ORCL000022. Finally, epicRealm has asserted in this case that the "releasing" of the Web server by the page server can occur simply by virtue of the fact that the page server receives the request and, as a result, implicitly releases (or frees) the Web server to process other requests. Although I disagree with epicRealm's implicit releasing theory, OWS 1.0 would disclose such releasing in the event the Court adopts epicRealm's theory because, according to epicRealm, the Oracle7 Server's receipt of the request would relieve the Web Listener of the obligation to process it.

wherein said routing step further includes the steps of intercepting said request at said Web server,

47. OWS 1.0 further discloses the step of intercepting said request at said Web server: "When the Oracle Web Listener receives a request from a client, it first determines whether that request is for a static document or a dynamic document. If the request is for a static document, the Web Listener sends the file and the associated type information directly to the client. If the request is for a dynamic document, it is created 'on the fly' by a program invoked by the Web Listener, in compliance with the Common Gateway Interface (CGI)." *Id.* Thus the Web

[8] I understand that epicRealm has also asserted an explicit releasing infringement theory under both parties' proposed "releasing" constructions. Under this theory, an accused page server "releases" the Web server to process other requests by sending a TCP acknowledgement message ("TCP ACK") to the Web server after receipt of the request (a.k.a., epicRealm's "TCP releasing" theory). EpicRealm contends that this TCP ACK acts to explicitly "release" the Web server. Similarly, in the Texas Actions, epicRealm has asserted another explicit releasing theory that involves Socket-Bind-Listen operating system calls. Although I disagree with epicRealm's explicit releasing theories because TCP ACKs and Socket-Bind-Listen calls do not release the Web server to process other requests, OWS 1.0 discloses such explicit releasing under both parties' proposed constructions.

Listener performs intercepting.[9] Particularly, under epicRealm's proposed construction for "intercepting," software running on the Web Listener intercepts the handling of dynamic Web page requests at the Web Listener and invokes another program (the Oracle Web Agent) to handle the request for the generation of the dynamic Web page. To the extent that epicRealm applies Oracle's proposed "intercepting" construction to Oracle's accused products in the same (or a similar) manner that it is applying its own proposed construction, the Web Listener performs intercepting under Oracle's proposed construction. In other words, to the extent that epicRealm interprets Oracle's proposed "intercepting" construction to mean that the processing of certain requests is interrupted at the Web server for the purpose of being handed off, OWS 1.0 discloses such intercepting.

routing said request from said Web server to a dispatcher, and dispatching said request to said page server;

48. OWS 1.0 discloses routing said request from said Web server (the Web Listener) to a dispatcher (an OWS Web Agent). The OWS Web agent is a "dispatcher" and performs "dispatching" under both parties' proposed constructions for those terms.[10] Particularly, the Oracle Web Agent is a CGI program that the Web Listener executes when a request is received for a dynamic document:

> The Oracle Web Agent is a program that is invoked by the Oracle Web Listener when a request for a database procedure is received. It handles the details of making a connection to the Oracle7 Server. A Web Agent will connect to a single Oracle7 Server, using a specific database username and password which are specified as part of a Web Agent service. To connect to different servers, or

---

[9] The parties' proposed constructions for "intercepting said request . . ." are listed in CHART 1 attached to Oracle's Motion for Summary Judgment of Invalidity.

[10] The parties' proposed constructions for "dispatcher" and "dispatching said request . . ." are listed in CHART 1 attached to Oracle's Motion for Summary Judgment of Invalidity.

> different schemas on the same server, multiple Web Agent services may be configured on a single Oracle WebServer.

Id. at ORCL000023. Thus the Web Agent analyzes a dynamic Web page request to make an informed choice of which Web Agent service is to be invoked to handle (i.e., process) the request. "When a request from a Web browser comes in, the Web Listener will extract the service name that is embedded in the URL and find out which parameters to use by reading the owa.cfg file." *Id.*. at ORCL000117. Thus, according to epicRealm's proposed construction for "dispatching," the Oracle Web Agent uses dynamic information stored in at least a configuration file to select the page server (an Oracle7 Server) that can more efficiently process the request. Because Oracle's proposed construction for "dispatching" is broader than that of epicRealm's, any prior art disclosure that meets epicRealm's construction will also meet Oracle's.

49. After an Oracle Web Agent dispatches a request to an Oracle7 Server, OWS 1.0 performs further dispatching of the request via SQL*Net dispatching. The OWS 1.0 User Guide makes the express statement that SQL*Net may optionally be used between the Web Agent and the Oracle7 Servers. *Id.* at ORCL000022. Additionally, the OWS 1.0 User Guide describes this as follows:

> To connect to the Oracle7 Server, the Web Agent needs the following information to be specified in the Web Agent service: • username • password • ORACLE_HOME • ORACLE_SID (for local databases only) • SQL*Net V2 Service Name or Connect String (for remote databases only). The Administration Utility allows the administrator to display, create, modify or delete a Web Agent service. With the Web Agent Creation form, you do not need to modify the configuration file for the Oracle Web Agent service (owa.cfg) directly. *Id.* at ORCL000117-119.

50. SQL*NET dispatching is also described in more detail in the reference "Understanding SQL*Net, Release 2.2," Part No. A32090-1 (Oracle Corp. 1995). Excerpted at A49. SQL*Net fundamentally allows the creation of distributed networked databases that reside

on disparate database servers (page servers), but that appear to the user as a single database: "SQL*Net allows Oracle tools and applications to access, manipulate, share and store data in Oracle databases residing on remote servers. In addition, SQL*Net enables data access between multiple database servers." *Id.* at ORCL01806623. Of course in a configuration having multiple page servers there must be a mechanism to decide which one should receive a request. Beginning at page ORCL01806663, the SQL*NET reference describes how dispatching occurs via a network listener. When the network listener receives a request it has the option to either (1) spawn a dedicated server and route the request to it, (2) route the request to a shared dispatcher of a multi-threaded server, or (3) route the request to one of the prespawned dedicated servers depending on the nature of the request and dynamic information maintained about the various servers. Figure 2-3 at ORCL01806664 [A49] illustrates this process:

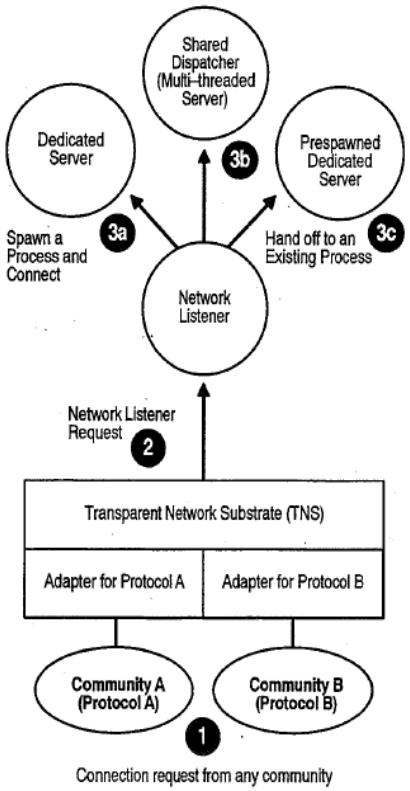


Figure 2 – 3 Network Listener in SQL*Net Connection

51. Beginning at page ORCL01806665, [A49] the SQL*NET reference describes how SQL*NET establishes connections and dispatches requests to particular prespawned dedicated servers. Additionally, at pages ORCL01806667-669, the SQL*Net reference describes how a request to a "multi-threaded server" is handled using shared dispatchers: "The listener receives the connection request, performs the connection handshake and determines if the client is allowed to connect … The listener issues a redirect message back to the client containing the address of the least-called dispatcher that is listening on the protocol used by the client … The listener and dispatcher perform a short handshake to update each other of the presence of a new connection. This is so that the listener can load balance connections between dispatchers." *Id.* at ORCL01806668. The request must be examined to determine which protocol it requires. The dynamic information maintained about page servers is their level of utilization (load). An informed selection is made based on this dynamic information, so SQL*Net performs dispatching under both parties' proposed constructions for that term.

52. Additionally, Release 2.3 of SQL*Net offered a more advanced form of load balancing. A technical description of Oracle's load balancing can be found in "Understanding SQL*Net, Release 2.3," Part No. A42484-1 (Oracle Corp. 1996), excerpted at Appendix, A50. *See*, generally, "Listener Load Balancing" *Id*. at ORCL01808646-647. "All dispatchers register with all the listeners listed in the database parameter file, and the client randomizes between the listeners. The listener passes the address of the least-used dispatcher to the client, and the client connects using that dispatcher." *Id.* at ORCL01808647. This method of load balancing implements "dispatching" under both parties' proposed constructions for that term.

processing said request, said processing being performed by said page server while said Web server concurrently processes said other requests; and

53. OWS 1.0 discloses this claim limitation as well. The Oracle7 Server (the page

server) processes the dynamic page generation request and returns the generated page to the Web Agent. A40 at ORCL000115-116. This process occurs simultaneously with the Web Listener (web server) processing other requests because the Web Listener operates asynchronously: "The Oracle Web Listener can handle a large number of simultaneous requests, and has advanced features which use system resources more efficiently than other HTTP servers available on the market." *Id.* at ORCL000022. Additionally, as discussed above, the Oracle7 Server's release of the Web Listener (the Web server) to process other requests (as asserted by epicRealm) naturally allows the Web Listener to concurrently process other requests at the same time as the Oracle7 Server (the page server). Additionally, OWS 1.0 allows for instantiation of multiple Web Listeners, allowing for concurrent processing of requests.

dynamically generating a Web page in response to said request, said Web page including data dynamically retrieved from one or more data sources.

54. OWS 1.0 discloses this final limitation as well. The Oracle7 Server (the page server) processes the dynamic page generation request and returns the generated Web page[11] to the Web Agent. See, e.g., *Id.* at ORCL000115-116. The generated web page includes data dynamically retrieved from one or more data sources by means of the PL/SQL procedures, which access an SQL database (i.e., an Oracle7 Server). *Id.*

Claim 2 of the '554 Patent - The computer-implemented method in claim 1 wherein said step of processing said request includes the step of identifying said one or more data sources from which to retrieve said data.

[11] The Web page generated by the OWS 1.0 page server meets both parties' proposed constructions for "Web page," namely, an "a file containing embedded commands in a Web formatting language such as HTML, capable of being displayed on a Web browser" as proposed by Oracle and, accordingly, "Web content displayable through a Web browser" as proposed by epicRealm.

55. OWS 1.0 discloses the step of identifying one or more data sources[12] from which to retrieve said data. Particularly, "[a]fter connecting to the database the Web Agent invokes the appropriate PL/SQL procedure, whose name is obtained by parsing the PATH_INFO environment variable." A40 at ORCL000115. The data sources to be used are identified by examining the PATH_INFO environment variable.

Claim 3 of the '554 Patent - The computer-implemented method in claim 2 wherein said step of dynamically generating said Web page includes the step of dynamically retrieving said data from said one or more data sources.

56. OWS 1.0 discloses dynamically retrieving said data from one or more data sources. Particularly, "[t]he Oracle WebServer is an HTTP server with a tightly integrated Oracle7 Server that enables the creation of dynamic HTML documents from data stored in an Oracle database. When the data changes, these HTML documents are updated automatically. . . . This approach supplements the presentation of static, or unchanging, data which is found on most sites today, with the dynamic real-time data present in business systems based on the Oracle7 Server." *Id.* at ORCL000021. The OWS 1.0 User's Guide further states: "The PL/SQL procedure executes, generating an HTML document. With the help of the Developer's Toolkit, the PL/SQL procedure extracts data from the Oracle7 database and generates an HTML document in a PL/SQL table." *Id.* at ORCL000116.

Claim 4 of the '554 Patent - The computer-implemented method in claim 3 wherein said step of processing said request includes the step of said page server maintaining a connection cache to said one or more data sources.

---

[12] The parties' proposed constructions for "data sources" are listed in CHART 1 attached to Oracle's Motion for Summary Judgment of Invalidity. I have applied these constructions, and epicRealm's application of the constructions, in my invalidity analysis with respect to OWS 1.0.

57. Connection caching[13] is used in SQL*Net, which may optionally be used with OWS 1.0. A49 (*Understanding SQL Net, Release 2.2)* at ORCL01806717. Particularly, as shown at ORCL01806717, the SQL*Net 2.2 Manual discloses that the Oracle7 Server keeps a store of information regarding connections to the database including (1) the type of service handlers, (2) the number of "established," "refused," "current," and "max," connections associated with those handlers, and (3) the "state" of those handlers, which can be used to affect subsequent connect times to data sources.

Claim 5 of the '554 patent - The computer-implemented method in claim 3 wherein said step of processing said request includes the step of logging into said one or more data sources.

58. OWS 1.0 discloses logging into[14] said one or more data sources. Particularly, the OWS 1.0 User's Guide states: "In order to connect to an Oracle7 Server, the Web Agent requires certain information, such as which server to connect to and what username and password to use." A40 at ORCL000115. The username and password are used to "log into" the data source according to Oracle's proposed construction for that term:

> A Web Agent will connect to a single Oracle7 Server, using a specific database username and password which are specified as part of a Web Agent service. To connect to different servers, or different schemas on the same server, multiple Web Agent services may be configured on a single Oracle WebServer. This allows for great flexibility in the creation of applications that unify data from several different servers, while controlling exactly what information a Web client can access. . . . When the Oracle Web

[13] The parties' proposed constructions for "connection cache" are listed in CHART 1 attached to Oracle's Motion for Summary Judgment of Invalidity. I have applied these constructions, and epicRealm's application of the constructions, in my invalidity analysis with respect to OWS 1.0.

[14] The parties' proposed constructions for "logging into" are listed in CHART 1 attached to Oracle's Motion for Summary Judgment of Invalidity. I have applied these constructions, and epicRealm's application of the constructions, in my invalidity analysis with respect to OWS 1.0.

> Agent logs into the Oracle7 Server, it starts a PL/SQL procedure that has been created by the user to generate an HTML page as its output. *Id.* at ORCL000023.

Claim 7 of the '554 patent - The computer-implemented method in claim 3 wherein said page server includes custom HTML extension templates for configuring said Web page.

59. OWS 1.0 discloses page servers including custom HTML extension templates for configuring a Web page. An extension template is a construct that is not HTML but that can easily be converted into HTML. It "extends" HTML by providing the user with additional facilities. OWS 1.0 provides a Developer's Toolkit incorporating hypertext procedures (HTP), described in Section 6 of the OWS 1.0 User's Guide: "A hypertext procedure generates a line in an HTML document that contains the HTML tag that corresponds to its name. For instance, the htp.anchor procedure generates an anchor tag." *Id.* at ORCL000132. A variety of hypertext procedures are provided with the toolkit, including "Body Tags," "List Tags," "Form Tags" and "Table Tags" which "allow the user to insert tables and manipulate the size and columns of the table in a document." *Id.* at ORCL000132-169. All of these are HTML extension templates.

Claim 8 of the '554 patent - The computer-implemented method in claim 7 wherein said step of processing said request further includes the step of inserting said dynamically retrieved data from said one or more data sources into said custom HTML extension templates.

60. OWS 1.0 discloses inserting said dynamically retrieved data from said one or more data sources into said custom HTML extension templates. The OWS 1.0 User's Guide gives many examples of inserting dynamic data obtained by accessing an SQL database into an HTML extension template. For example, the procedure htp.cite, described on page ORCL000151 of A40, when called with two arguments, e.g. htp.cite(ctext, cattributes), causes the arguments to be substituted into a template as follows: <CITE cattributes>ctext</CITE>. See PL/SQL procedures at *Id.* at ORCL000124-126.

Claim 9 of the '554 patent

A networked system for managing a dynamic Web page generation request, said system comprising:

61. Because OWS 1.0 operated with a Web server it was networked as shown throughout the OWS 1.0 User's Guide. A40.

one or more data sources;

62. OWS 1.0 discloses a networked system with one or more data sources. For example, the OWS 1.0 User's Guide discloses that the generated web page includes data dynamically retrieved from one or more data sources by means of the PL/SQL procedures, which access an SQL database: "To connect to different servers, or different schemas on the same server, multiple Web Agent services may be configured on a single Oracle WebServer. This allows for great flexibility in the creation of applications that unify data from several different servers, while controlling exactly what information a Web client can access." *Id.* at ORCL000023.

a page server having a processing means;

63. As discussed above with respect to claim 1, OWS 1.0 discloses a page server having a processing means. For example, the OWS 1.0 User's Guide states: "The Oracle7 Server provides the storage for all dynamic data in relational tables, and all the program logic used to create dynamic HTML pages." *Id.* at ORCL000021. Thus, as discussed with respect to claim 1 above, the Oracle7 Server is a "page server" of the patent and, accordingly, contains a "page server processing means" under both parties' proposed constructions for that term.[15]

[15] The parties' proposed constructions for "page server processing means" are listed in CHART 1 attached to Oracle's Motion for Summary Judgment of Invalidity. I have applied these constructions, and epicRealm's application of the constructions, in my invalidity analysis with respect to OWS 1.0.

Particularly, the Oracle7 Server is a "page server" under epicRealm's proposed constructions because, as discussed above, it includes software for generating "dynamic HTML pages." *Id.* Similarly, the Oracle7 Server is a "page server" under Oracle's proposed construction for that term because, as discussed above, ORCL000021 discloses the server running on a different machine (i.e., different processor) from that of the Web Server (the Web Listener). Further, a processor of a computer that runs an Oracle7 Server is a page server processing means.

a first computer system including means for generating said request; and

64. OWS 1.0 discloses a first computer system including means for generating said request.[16] The first computer system of OWS 1.0 is a client computer that has a processor that runs a web browser as shown on page *Id.* at ORCL000022. This client computer generates Web page requests.

a second computer system including means for receiving said request from said first computer,

65. OWS 1.0 discloses a second computer system including means for receiving said request from said first computer.[17] The second computer system is a computer having a processor that runs the Web Listener as shown in a rectangular box on page ORCL000022 of A40. Again, the Oracle7 Server is a "page server" of the epicRealm patents. The dynamic page generation request is received by the Web Listener (the Web server) and routed to the Oracle7

[16] The parties' proposed constructions for "means for generating said request" are listed in CHART 1 attached to Oracle's Motion for Summary Judgment of Invalidity. I have applied these constructions, and epicRealm's application of the constructions, in my invalidity analysis with respect to OWS 1.0.

[17] The parties' proposed constructions for "means for receiving said request" are listed in CHART 1 attached to Oracle's Motion for Summary Judgment of Invalidity. I have applied these constructions, and epicRealm's application of the constructions, in my invalidity analysis with respect to OWS 1.0.

Server, as shown in the diagram on page ORCL000022. The means is a processor of a computer that runs the Web Listener.

said second computer system also including a router, said router routing said request from said second computer system to said page server,

66. As discussed above, OWS 1.0 discloses a second computer system including means for receiving said request from said first computer. That second computer system includes a router[18] as shown on page ORCL000022 of A40. The second computer system of the figure on ORCL000022 (including the Web Listener) routes requests to an Oracle7 Server (the page server). As discussed above, the Web Listener can spawn a Web Agent process (also part of the router), which performs the claimed step of dispatching to an Oracle7 Server.

wherein said routing further includes intercepting said request at said second computer, routing said request from said second computer to a dispatcher, and dispatching said request to said page server

67. As discussed above with respect to claim 1 of the '554 patent, OWS 1.0 discloses these claim limitations including "intercepting said request at said second computer," "routing said request from said second computer to a dispatcher," and dispatching said request to said page server." These claim limitations are identical to that of claim 1 with the exception of the "second computer" being used in place of the "Web server" of claim 1. Because the OWS 1.0 Web Listener (the Web server of claim 1) runs on a second computer (i.e., a computer different from that of the client computer), OWS 1.0 anticipates these claim limitations in the same way it anticipates claim 1.

---

[18] The parties' proposed constructions for "router" are listed in CHART 1 attached to Oracle's Motion for Summary Judgment of Invalidity. I have applied these constructions, and epicRealm's application of the constructions, in my invalidity analysis with respect to OWS 1.0.

said page server receiving said request and releasing said second computer system to process other requests,

68. As discussed above with respect to claim 1 of the '554 patent, OWS 1.0 discloses this claim limitation including "said page server receiving said request and releasing said second computer system to process other requests." This claim limitation is identical to that of claim 1 with the exception of the "second computer" being used in place of the "Web server" of claim 1. Because the OWS 1.0 Web Listener (the Web server of claim 1) runs on a second computer (i.e., a computer different from that of the client computer), OWS 1.0 anticipates this claim limitation in the same way it anticipates claim 1.

said page server processing means processing said request and dynamically generating a Web page in response to said request, said Web page including data dynamically retrieved from said one or more data sources.

69. As discussed above with respect to claim 1 of the '554 patent, OWS 1.0 discloses these claim limitations including "intercepting said request at said second computer," "routing said request from said second computer to a dispatcher," and dispatching said request to said page server." These claim limitations are identical to that of claim 1 with the exception of the "second computer" being used in place of the "Web server" of claim 1. Because the OWS 1.0 Web Listener (the Web server of claim 1) runs on a second computer (i.e., a computer different from that of the client computer), OWS 1.0 anticipates these claim limitations in the same way it anticipates claim 1.

Claim 10 of the '554 patent - The networked system in claim 9 wherein said router in said second computer system includes:

70. As discussed above, OWS 1.0 operates on a network.

an interceptor intercepting said request at said second computer system and routing said request; and

71. As discussed above with respect to claim 1 of the '554 patent, OWS 1.0 discloses this claim limitation including "an interceptor intercepting said request at said second computer

system and routing said request." Because OWS 1.0 discloses the step of intercepting a request at a Web server, the OWS Web Listener (the Web server) includes software for performing that act (i.e., an interceptor).

a dispatcher receiving said routed request from said interceptor and dispatching said request to said page server.

72. As discussed above with respect to claim 1 of the '554 patent, OWS 1.0 discloses this claim limitation including "a dispatcher receiving said routed request from said interceptor and dispatching said request to said page server." Particularly, the OWS 1.0 interceptor of the Web Listener (the Web server) routes request to the OWS 1.0 Web Agent (the dispatcher) for dispatching to a page server.

Claim 11 of the '554 patent

A machine readable medium having stored thereon data representing sequences of instructions, which when executed by a computer system, cause said computer system to perform the steps of:

73. OWS 1.0 is a programmed system whose instructions are on a machine readable medium.[19]

routing a dynamic Web page generation request from a Web server to a page server, said page server receiving said request and releasing said Web server to process other requests wherein said routing step further includes the steps of intercepting said request at said Web server, routing said request from said Web server to a dispatcher, and dispatching said request to said page server;

processing said request, said processing being performed by said page server while said Web server concurrently processes said other requests; and

dynamically generating a Web page, said Web page including data retrieved from one or more data sources.

[19] The parties' proposed constructions for "machine readable medium" are listed in CHART 1 attached to Oracle's Motion for Summary Judgment of Invalidity. I have applied these constructions, and epicRealm's application of the constructions, in my invalidity analysis with respect to OWS 1.0.

74. As discussed above with respect to claim 1 of the '554 patent, OWS 1.0 discloses these claim limitations, which are substantially identical to the limitations of claim 1.

Claim 2 of the '335 patent (including limitations of Claim 1)

A computer-implemented method for managing a dynamic Web page generation request to a Web server, said computer-implemented method comprising the steps of:

routing a request from a Web server to a page server, said page server receiving said request and releasing said Web server to process other requests wherein said routing step further includes the steps of:

intercepting said request at said Web server and routing said request to said page server;

processing said request, said processing being performed by said page server while said Web server concurrently processes said other requests; and

dynamically generating a Web page in response to said request, said Web page including data dynamically retrieved from one or more data sources.

75. As discussed above with respect to claim 1 of the '554 patent, OWS 1.0 discloses these claim limitations, which are substantially identical to the limitations of claim 1.

Claim 2 of the 335 patent - The computer-implemented method in claim 1 wherein said step of routing said request includes the steps of:

routing said request from said Web server to a dispatcher; and

dispatching said request to said page server.

76. As discussed above with respect to claim 1 of the '554 patent, OWS 1.0 discloses these claim limitations, which are substantially identical to the limitations of claim 1.

Claim 16 of the '335 patent (including limitations of claim 15)

A computer-implemented method comprising the steps of:

transferring a request from an HTTP-compliant device to a page server, said page server receiving said request and releasing said HTTP-compliant device to process other requests wherein said transferring step further includes the steps of:

intercepting said request at said HTTP-compliant device and transferring said request to said page server;

processing said request, said processing being performed by said page server while said HTTP-compliant device concurrently processes said other requests; and

dynamically generating a page in response to said request, said page including data dynamically retrieved from one or more data sources.

77. As discussed above with respect to claim 1 of the '554 patent, OWS 1.0 discloses these claim limitations, which are substantially identical to the limitations of claim 1. The only significant difference between the limitations of claim 1 of the '554 patent and this claim is that the term "HTTP-compliant device" of this claim is used in place of "Web server" of claim 1. As discussed above, the Oracle WebServer is an HTTP-compliant device: "The Oracle WebServer is an HTTP server with a tightly integrated Oracle7 server that enables the creation of dynamic HTML documents from data stored in an Oracle database." A40 at ORCL000021. "The Oracle Web Listener can handle a large number of simultaneous requests, and has advanced features which use system resources more efficiently than other HTTP servers available on the market." *Id.* at ORCL000022.

Claim 16 of the '335 patent - The computer-implemented method in claim 15 wherein said step of transferring said request includes the steps of:

transferring said request from said HTTP-compliant device to a dispatcher; and

dispatching said request to said page server.

78. As discussed above with respect to claim 1 of the '554 patent, OWS 1.0 discloses these claim limitations, which are substantially identical to the claim limitations of claim 1.

B. All the Asserted Claims Are Anticipated by Oracle WebServer 2.0

79. As demonstrated below, the Oracle Web Server 2.0 system (including the Oracle7 server) (referred to below as "OWS 2.0") anticipates all of the Asserted Claims of the epicRealm patents.

Claim 1 of the '554 patent

A computer-implemented method for managing a dynamic Web page generation request to a Web server, said computer-implemented method comprising the steps of:

80. As stated in the Oracle WebServer ("OWS") 2.0 User's Guide (Part No. A236646-1), OWS 2.0 managed dynamic Web page generation requests to a Web server (the HTTP Server

or Web Listener): "Using the Oracle Web Listener, your web site can respond to client requests by generating HTML documents dynamically." A42 at ORCL01816648. Additionally, the "database is used for Web pages that are generated at runtime using 'live' data." *Id.* at ORCL01816643.

routing said request from said Web server to a page server, said page server receiving said request and releasing said Web server to process other requests,

81. OWS 2.0 discloses routing a dynamic Web page generation request[20] from a Web server (the "HTTP Server" or "Web Listener" of OWS 2.0)[21] to a page server (including a Web Request Broker Executable Engine ("WRBX") and WRB Service). Particularly, a Technical Note describing OWS 2.0 states: "The HTTP Server is the network layer component of the Oracle WebServer. It listens for incoming HTTP requests, delivers static files and runs simple CGI programs, and hands off everything else to the Web Request Broker (WRB), where a request is handled by a server extension." A44 at ORCL0000764. The Web Listener (the Web server) of OWS 2.0 is further described in OWS 2.0 User's Guide at A42 (*OWS 2.0)* at ORCL01816643-648. Particularly, the User's Guide states: "The Web Listener is the component that receives a URL from a Web browser and sends back the appropriate output" and "[u]sing the Oracle Web Listener, your web site can respond to client requests by generating HTML document dynamically." *Id.* at ORCL01816643 and 648.

---

[20] This "request" meets both parties' proposed constructions for that term, namely, "a message that asks for a Web page specified by a URL" as proposed by Oracle and, accordingly, "a message that asks for a Web page" as proposed by epicRealm.

[21] The OWS 2.0 Web Listener meets both parties' proposed constructions for "Web server," namely, an "HTTP-compliant server software, or a machine running such software, that receives Web page requests and returns Web pages in response to the requests" as proposed by Oracle and, accordingly, "software, or a machine having software, that receives Web page requests and returns Web pages in response to the requests" as proposed by epicRealm.

82. The Web Listener is also responsible for routing particular dynamic Web page requests to a page server by means of a WRB Dispatcher. After the Web Listener hands of a requests to the WRB, the WRB in turn dispatches requests to WRBX processes that receive the requests and invoke server extensions called WRB Services (part of the page servers). *Id.* at ORCL01816643 and ORCL01816652-653. These WRB Services include, for example, the "PL/SQL Agent, the Java Interpreter, and the LiveHTML Interpreter," and are used to generate dynamic Web pages. *Id.* at ORCL01816808 and ORCL01816652-653. The WRBX processes (i.e., instances of WRB Services) are the "page servers" of the epicRealm patents under both parties' proposed constructions for that term.[22] Particularly, a WRBX process is a "page server" under epicRealm's proposed constructions because, as discussed above, it includes software for generating "dynamic HTML pages." Similarly, a WRBX process is a "page server" under Oracle's proposed construction for that term because the OWS 2.0 User's Guide discloses, for example, that the PL/SQL Agent can execute stored PL/SQL procedures on a database running on a different machine (i.e., different processor) from that of the Web Server (the Web Listener) just as in the case of OWS 1.0.[23] *Id.* at ORCL01816653-654.

83. Additionally, OWS 2.0 discloses "releasing said Web Server to process other requests" under both parties' proposed constructions for that claim term.[24] For example, as stated in the OWS 2.0 Technical Note: "All inter-process communication is handled by the transport

---

[22] The parties' proposed constructions for "page server" are listed in CHART 1 attached to Oracle's Motion for Summary Judgment of Invalidity.

[23] Alternatively, it would be obvious to one of ordinary skill in the art to run the WRBXs (the page servers) on one or more different machines (i.e., different processors) from that of the Web server because such a configuration would spread out the processing of requests across multiple processors and, as a result, would increase the efficiencies of the request processing.

[24] The parties' proposed constructions for "page server . . . releasing said Web server . . ." are listed in CHART 1 attached to Oracle's Motion for Summary Judgment of Invalidity.

independent WRB protocol. WebServer 2.0 supports standard IPC mechanisms." A44 (*OWS 2.0 Technical Note)* at ORCL000766. A standard IPC mechanism is Socket-Bind-Listen, hence SBL releasing is performed according to EpicRealm's SBL releasing theory. See explanation at fn.8 above.

84. Additionally, Oracle 2.0 documentation discloses a diagram at A51 (*Web Request Broker API)* at ORCL01606867 showing that a Web Application (such as PL/SQL Agent) sends a REQ COMPLETE Response message, which is an act that is separate from merely receiving the request. Under epicRealm's explicit releasing theory,[25] with which I do not agree, the HTTP Server receives the REQ COMPLETE Response message and frees up certain resources that would either allow processing of other requests or facilitate more efficient processing of already received requests. The HTTP Server (Web server) must free such resources because otherwise the REQ COMPLETE Response message has no purpose. *Id.* at ORCL01606867. Additionally, under epicRealm's SBL releasing theory, the INIT Response message sent from a Web Application (such as PL/SQL Agent) to the HTTP Server is an action, which is separate from merely receiving the request that informs the HTTP Server that it is now able to offload the processing of the request to the now initialized WEB Application, prospectively freeing up the HTTP Server to process other requests once a request capable of being processed by the WEB Application is received. Under epicRealm's infringement theories with respect to the claimed act of "releasing the Web server," with which I disagree, all of the above-described actions would meet both parties' proposed constructions for "releasing."

85. Finally, epicRealm has asserted in this case that the "releasing" of the Web server

---

[25] See explanation at fn. 8 above.

by the page server can occur simply by virtue of the fact that the page server receives the request and, as a result, implicitly releases (or frees) the Web server to process other requests. Although I disagree with this implicit releasing theory, OWS 2.0 would disclose such releasing in the event the Court adopts epicRealm's theory because, according to epicRealm, a WRBX's receipt of the request would relieve the Web Listener of the obligation to process it.

wherein said routing step further includes the steps of intercepting said request at said Web server,

86. OWS 2.0 further discloses the step of intercepting said request at said Web server: "The HTTP engine [the Web server] does not make any attempt at interpreting the URL, instead it is immediately handed off to the Web Request Broker for further processing." A44 (*OWS 2.0 Technical Note)* at ORCL000766. This hand off is further described as: "The HTTP Server . . . listens for incoming HTTP requests, delivers static files and runs simple CGI programs and hands off everything else to the Web Request Broker (WRB), where a request is handled by a server extension." *Id.* at ORCL000764; see also A42 (*OWS User's Guide Release 2.0)* at ORCL01816643.

87. For example, the Web Request Broker may be configured to receive and dispatch all URL requests beginning with /java to the integrated Java Interpreter. These requests are intercepted and routed to the WRB for handling. "However, if no suitable WRB Service is found, the request is passed back to the Web Listener and standard processing continues." A44 at ORCL0000766. Thus the Web Listener performs intercepting.[26] Particularly, under epicRealm's proposed construction, software running on the OWS 2.0 HTTP server can intercept

[26] The parties' proposed constructions for "intercepting said request . . ." are listed in CHART 1 attached to Oracle's Motion for Summary Judgment of Invalidity.

the handling of a dynamic Web page request intended for the WRB and will route it to the WRB for handling (i.e., dispatching to a page server). To the extent that epicRealm applies Oracle's proposed "intercepting" construction to Oracle's accused products in the same (or a similar) manner that it is applying its own proposed construction, the Web Listener performs intercepting under Oracle's proposed construction. In other words, to the extent that epicRealm interprets Oracle's proposed "intercepting" construction to mean that the processing of certain requests is interrupted at the Web server for the purpose of being handed off, OWS 2.0 discloses such intercepting.

routing said request from said Web server to a dispatcher, and dispatching said request to said page server;

88. OWS 2.0 discloses routing said request from said Web server (the Web Listener) to a dispatcher (the WRB Dispatcher). The OWS WRB Dispatcher is a "dispatcher" and performs "dispatching" under both parties' proposed constructions for those terms.[27] Particularly, "[t]he HTTP Server . . . listens for incoming HTTP requests, delivers static files and runs simple CGI programs, and hands off everything else to the Web Request Broker (WRB), where a request is handled by a server extension." *Id.* at ORCL000764. First, "[t]he WRB dispatcher must decide what type of object is being requested. To do this, it examines the WRB configuration file, which maps virtual directories to WRB Services." *Id.* at ORCL000766; see also A42 (*OWS User's Guide Release 2.0)* at ORCL01816652-653.

89. As discussed above, the WRB Dispatcher then dispatches requests to a WRBX (i.e., an instance of a WRB Service) which is "[o]ne of a pool of processes that the WRB

[27] The parties' proposed constructions for "dispatcher" and "dispatching said request . . ." are listed in CHART 1 attached to Oracle's Motion for Summary Judgment of Invalidity.

maintains continuously, so that HTTP requests requiring the execution of programs are not slowed down by the performance cost of spawning a new process." *Id.* at ORCL01816808. "WRBX's are associated with WRB Cartridges and are created and destroyed according to workload." *Id.* When selecting a particular WRBX to process a request, "[t]he WRB Dispatcher performs dynamic load balancing between multiple instances of a WRB Service, and the webmaster can configure each WRB Service to have a minimum and maximum number of instances. An instance corresponds to a process in a UNIX environment." A44 (*OWS 2.0 Technical Note)* at ORCL0000766; see also A42 (*OWS User's Guide Release 2.0)* at ORCL01816653. Accordingly, the WRB Dispatcher dispatches requests according to epicRealm's proposed construction for that term because it examines a request to make an informed selection of which page server (a WRBX process) should process the request based on dynamic information maintained about page servers (i.e., which WRBX is free and which one is configured to run the desired service), which is information indicating which page server can more efficiently process the request. Because Oracle's proposed construction for "dispatching" is broader than that of epicRealm, OWS 2.0 must also perform dispatching under Oracle's proposed construction.

processing said request, said processing being performed by said page server while said Web server concurrently processes said other requests; and

90. OWS 2.0 discloses this claim limitation as well. Concurrent processing between the Web server and the page server is taught expressly in the OWS 2.0 User's Guide: "When the Web Listener receives a URL, it determines whether the request requires the use of a Service to be accessed through the Web Request Broker (WRB), a program to be accessed through the CGI interface, or whether access to the file system of the machine on which the Listener resides is sufficient. If WRB access is required, the Listener passes the request to WRB Dispatcher for

processing: then it returns to the task of listening for more incoming HTTP requests." *Id.* at ORCL01816643; see also A44 (*OWS 2.0 Technical Note)* at ORCL000764. Because the Listener proceeds to listen for and process other requests and does not wait for the WRB Dispatcher to finish, it is capable of concurrently processing requests. Additionally, another OWS 2.0 document confirms that this claim limitation is met: "Oracle's HTTP Server is designed to be a very scalable, high-performance network daemon. It is implemented as a multi-threaded, single-process asynchronous engine, allowing multiple concurrent requests to be processed at the same time using standard HTTP or HTTP over SSL." *Id.* at ORCL000764.

dynamically generating a Web page in response to said request, said Web page including data dynamically retrieved from one or more data sources.

91. OWS 2.0 discloses this final limitation as well. As discussed above, the WRB Service instances (including the PL/SQL Agent, the Java Interpreter or the LiveHTML Interpreter) process the dynamic page generation requests and return the generated Web page[28] to the Web server. See, e.g., A42 at ORCL01816653-657. "As documented in the CGI 1.1 specification, the Oracle Web Listener takes the contents of standard output and returns it to the Web browser that requested the dynamic HTML document." *Id.* at ORCL01816707.

92. For example, "QUERY_FORM prints an HTML page with all the columns for the specified table. Invoke the procedure from a Web Browser with a URL like: http://yourhost:port_num/service_name/owa/query_form?the_table=emp . . . Invoking this procedure brings up a page that looks like [page shown at ORCL01816677]." *Id.* at

[28] The Web page generated by the OWS 2.0 page server meets both parties' proposed constructions for "Web page," namely, an "a file containing embedded commands in a Web formatting language such as HTML, capable of being displayed on a Web browser" as proposed by Oracle and, accordingly, "Web content displayable through a Web browser" as proposed by epicRealm.

ORCL01816676-677 and ORCL01816678-79.

93. Additionally, a Web page generated by OWS 2.0 includes data dynamically retrieved from one or more data sources. For example, as taught in the OWS 2.0 User's Guide, "[t]he Oracle WebServer . . . draws on information from the database and the operating system's (0S) file system as necessary to respond to the request. The file system can be used for static (hardcoded) Web pages, or form scripts that do not access the database, and the database is used for Web pages that are generated at runtime using 'live' data." *Id.* at ORCL01816643. Additionally, "LiveHTML code is formatted as SGML comments, so that it is ignored should the file ever find its way to the browser unparsed. The format for LiveHTML codes, therefore, is the following: <!--#command tag1="value1" tag2="value2" --> The tags are arguments to the commands, most of which actually only accept one of the possible tags. The possible commands and their associated tags are as follows: . . . include. This command specifies that a file is to be included in the generated HTML page at this point." A44 (*OWS 2.0 Technical Note)* at ORCL0000778-779 see also A42 (*OWS User's Guide Release 2.0)* at ORCL01816697-699.

Claim 2 of the '554 Patent - The computer-implemented method in claim 1 wherein said step of processing said request includes the step of identifying said one or more data sources from which to retrieve said data.

94. OWS 2.0 discloses the step of identifying one or more data sources[29] from which to retrieve said data. For example, "[t]he PL/SQL Agent uses Database Connection Descriptors (DCDs) to control the privileges an application runs under and the database schema to which it connects in a generalized and application-independent way." *Id.* at ORCL01816683.

---

[29] The parties' proposed constructions for "data sources" are listed in CHART 1 attached to Oracle's Motion for Summary Judgment of Invalidity. I have applied these constructions, and epicRealm's application of the constructions, in my invalidity analysis with respect to OWS 2.0.

Additionally, another OWS 2.0 document confirms this: the "[PL/SQL Agent] also needs to have access to all the details required to actually establish a permanent connection: typically a set of environment variables (ORACLE_HOME and ORACLE_SID for UNIX systems) and perhaps a SQL*Net connect descriptor. All of this information is referred to as a DCD (Database Connection Descriptor), and all of the DCDs known to the PL/SQL Agent are stored in a configuration file named owa.cfg (DCDs where known as Web Agent Services in Webserver 1.0). . . . DCDs are defined in the PL/SQL Agent's configuration file, but they are activated by creating a directory mapping in the Web Request Broker's configuration file." A44 at ORCL000772; see also A42 at ORCL01816672-682.

Claim 3 of the '554 Patent - The computer-implemented method in claim 2 wherein said step of dynamically generating said Web page includes the step of dynamically retrieving said data from said one or more data sources.

95. OWS 2.0 discloses dynamically retrieving said data from one or more data sources. For example, as taught in the OWS 2.0 User's Guide, "[t]he Oracle WebServer . . . draws on information from the database and the operating system's (0S) file system as necessary to respond to the request. The file system can be used for static (hardcoded) Web pages, or for CGI scripts that do not access the database, and the database is used for Web pages that are generated at runtime using 'live' data." A42 at ORCL01816643. Additionally, OWS 2.0 meets this limitation as described below with respect to the use of LiveHTML:

> LiveHTML code is formatted as SGML comments, so that it is ignored should the file ever find its way to the browser unparsed. The format for LiveHTML codes, therefore, is the following:
>
> <!--#command tag1="value1" tag2="value2" -->
>
> The tags are arguments to the commands, most of which actually only accept one of the possible tags. The possible commands and their associated tags are as follows: . . . include. This command specifies that a file is to be included in the generated HTML page at this point.

A44 at ORCL000778-779; see also A42 at ORCL01816697-699.

Claim 4 of the '554 Patent - The computer-implemented method in claim 3 wherein said step of processing said request includes the step of said page server maintaining a connection cache to said one or more data sources.

96. OWS 2.0 discloses connection caching.[30] Particularly, "[d]ue to the new architecture in Oracle Webserver 2.0 with the Web Request Broker, the PL/SQL Agent is roughly an order of magnitude faster than the previous version. This is mainly due to the fact that each instance of the PL/SQL Agent stays connected to Oracle7 between requests, and thus does not need to establish a new database connection each time a stored procedure needs to be executed." A44 at ORCL000771. This form of connection caching is further described as follows: "[a]t the lowest level, Oracle's internal API allows us to establish a connection to an Oracle7 Server using SQL*Net in one step, and actually logging on to the database in a separate step. Since each HTTP request needs a new session, we are technically logging on and off between requests, but this is still extremely fast since the Oracle7 Server connection is maintained between requests." *Id.* at ORCL000772.

Claim 5 of the '554 patent - The computer-implemented method in claim 3 wherein said step of processing said request includes the step of logging into said one or more data sources.

97. OWS 2.0 discloses logging into[31] said one or more data sources: "Oracle's internal API allows us to establish a connection to an Oracle7 Server using SQL*Net in one step,

---

[30] The parties' proposed constructions for "connection cache" are listed in CHART 1 attached to Oracle's Motion for Summary Judgment of Invalidity. I have applied these constructions, and epicRealm's application of the constructions, in my invalidity analysis with respect to OWS 2.0.

[31] The parties' proposed constructions for "logging into" are listed in CHART 1 attached to Oracle's Motion for Summary Judgment of Invalidity. I have applied these constructions, and epicRealm's application of the constructions, in my invalidity analysis with respect to OWS 2.0.

and actually logging on to the database in a separate step." *Id.* This form of logging into the database can involve presenting credentials for permission to access the database. For example, a URL that invokes the PL/SQL Agent must specify a DCD (Database Connection Descriptor). This is an OS file maintained by the WebServer that provides the username, password, database, and other information to be used to establish the database connection. A42 at ORCL01816653-654.

Claim 7 of the '554 patent - The computer-implemented method in claim 3 wherein said page server includes custom HTML extension templates for configuring said Web page.

98. OWS 2.0 discloses page servers including custom HTML extension templates for configuring a Web page. For example, the OWS 2.0 User's Guide discloses such extension templates with the use of LiveHTML: "LiveHTML. A way to embed dynamic content in Web pages. This content can be either other Web pages or the output of scripts run by the Operating System. LiveHTML is an Oracle extension of the NCSA standard Server Side Includes functionality." *Id.* at ORCL01816643. "The LiveHTML Interpreter enables you to include dynamic content in otherwise static Web pages." *Id.* at ORCL01816656. "LiveHTML code is formatted as SGML comments . . . . The format for LiveHTML codes, therefore, is the following: < ! -- #command tag1="value1" tag2="value2" -- >. The tags are arguments to the commands, most of which actually only accept one of the possible tags." *Id.* at ORCL01816697. "LiveHTML files supplement HTML with instructions that WebServer executes before transmitting the page. These instructions specify material that is to be included in the generated page. Said material can include other Web pages, environment variables, and the output of programs executed on the Server." *Id.* at ORCL01816803.

Claim 8 of the '554 patent - The computer-implemented method in claim 7 wherein said step of processing said request further includes the step of inserting said dynamically retrieved data from said one or more data sources into said custom HTML extension templates.

99. OWS 2.0 discloses inserting said dynamically retrieved data from said one or more data sources into said custom HTML extension templates. For example, the OWS 2.0 User's Guide discloses this with respect to LiveHTML: "include. This command specifies that a file is to be included in the generated HTML page at this point. The file can any of the following: - another LiveHTML file like the current one. - a regular HTML file. - an ASCII file." *Id.* at ORCL01816698.

Claim 9 of the '554 patent

A networked system for managing a dynamic Web page generation request, said system comprising:

100. Because OWS 2.0 operated with a Web server it was networked as shown throughout the OWS 2.0 User's Guide.

one or more data sources;

101. As discussed above with respect to claim 1, OWS 2.0 discloses a networked system with one or more data sources. Particularly, the OWS 2.0 User's Guide states: "[t]he Oracle WebServer . . . draws on information from the database and the operating system's (0S) file system as necessary to respond to the request. The file system can be used for static (hardcoded) Web pages, or for CGI scripts that do not access the database, and the database is used for Web pages that are generated at runtime using 'live' data." *Id.* at ORCL01816643. Additionally, "LiveHTML code is formatted as SGML comments, so that it is ignored should the file ever find its way t o the browser unparsed. The format for LiveHTML codes, therefore, is the following: <!--#command tag1="value1" tag2="value2" --> The tags are arguments to the commands, most of which actually only accept one of the possible tags. The possible commands and their associated tags are as follows: . . . include. This command specifies that a file is to be included in the generated HTML page at this point." A44 at ORCL0000778-779 see also A42 at

ORCL01816697-699.

a page server having a processing means;

102. As discussed above with respect to claim 1, OWS 2.0 discloses a page server having a processing means. Particularly, a Technical Note describing OWS 2.0 states: "The HTTP Server is the network layer component of the Oracle WebServer. It listens for incoming HTTP requests, delivers static files and runs simple CGI programs and hands off everything else to the Web Request Broker (WRB), where a request is handled by a server extension." A44 at ORCL0000764. The Web Listener (the Web server) of OWS 2.0 is further described in OWS 2.0 User's Guide at A42 at ORCL01816643-648. Particularly, the User's Guide states: "The Web Listener is the component that receives a URL from a Web browser and sends back the appropriate output" and "[u]sing the Oracle Web Listener, your web site can respond to client requests by generating HTML document dynamically." *Id.*

103. The Web Listener is also responsible for routing particular dynamic Web page requests to a page server by means of a WRB Dispatcher. After the Web Listener hands of a requests to the WRB, the WRB in turn dispatches requests to WRBX processes that receive the requests and invoke server extensions called WRB Services (the page servers). *Id.* at ORCL01816643 and ORCL01816652-653. These WRB Services include, for example, the "PL/SQL Agent, the Java Interpreter, and the LiveHTML Interpreter," and are used to generate dynamic Web pages. *Id.* at ORCL01816808 and ORCL01816652-653. As discussed above with respect to claim 1, the WRBX processes (i.e., instances of WRB Services) are the "page servers" of the epicRealm patents under both parties' proposed constructions for that term. The processors on the computer(s) that run these WRBX processes are the "page server processing

means" under both parties' proposed constructions for that term.[32] Further, as discussed above with respect to OWS 1.0, a processor of a computer that runs an Oracle7 Server is a page server processing means.

a first computer system including means for generating said request; and

104. OWS 2.0 discloses a first computer system including means for generating said request.[33] The first computer system of OWS 2.0 is a client computer that has a processor that runs a web browser: "from a web browser's point of view, Oracle WebServer 2.0 looks just like any other web server." A44 at ORCL000763. Also, the first computer system is the computer that has a processor that runs the web browser discussed in the OWS 2.0 User's Guide. A42 at ORCL01816644. This client computer generates Web page requests.

a second computer system including means for receiving said request from said first computer,

105. OWS 1.0 discloses a second computer system including means for receiving said request from said first computer.[34] The second computer system includes the OWS 2.0 Web Listener (the Web server), the Web Request Broker (the dispatcher), as shown on A44 at ORCL0000763, and the WRB Executable Engines (page servers).

---

[32] The parties' proposed constructions for "page server processing means" are listed in CHART 1 attached to Oracle's Motion for Summary Judgment of Invalidity. I have applied these constructions, and epicRealm's application of the constructions, in my invalidity analysis with respect to OWS 2.0.

[33] The parties' proposed constructions for "means for generating said request" are listed in CHART 1 attached to Oracle's Motion for Summary Judgment of Invalidity. I have applied these constructions, and epicRealm's application of the constructions, in my invalidity analysis with respect to OWS 2.0.

[34] The parties' proposed constructions for "means for receiving said request" are listed in CHART 1 attached to Oracle's Motion for Summary Judgment of Invalidity. I have applied these constructions, and epicRealm's application of the constructions, in my invalidity analysis with respect to OWS 2.0.

said second computer system also including a router, said router routing said request from said second computer system to said page server,

106. As discussed above with respect to claim 1 of the '554 patent, OWS 2.0 discloses a second computer system including means for receiving said request from said first computer. That second computer system includes a router[35] (including the HTTP Server and Web Request Broker) which, as discussed above, performs the functions of intercepting requests at the Web server and dispatching them to page servers (the WRBXs).

wherein said routing further includes intercepting said request at said second computer, routing said request from said second computer to a dispatcher, and dispatching said request to said page server

107. As discussed above with respect to claim 1 of the '554 patent, OWS 2.0 discloses these claim limitations including "intercepting said request at said second computer," "routing said request from said second computer to a dispatcher," and dispatching said request to said page server." These claim limitations are identical to that of claim 1 with the exception of the "second computer" being used in place of the "Web server" of claim 1. Because the OWS 2.0 HTTP Server (the Web server of claim 1) runs on a second computer (i.e., a computer different from that of the client computer), OWS 2.0 anticipates these claim limitations in the same way it anticipates claim 1.

said page server receiving said request and releasing said second computer system to process other requests,

108. As discussed above with respect to claim 1 of the '554 patent, OWS 2.0 discloses this claim limitation including "said page server receiving said request and releasing said second

[35] The parties' proposed constructions for "router" are listed in CHART 1 attached to Oracle's Motion for Summary Judgment of Invalidity. I have applied these constructions, and epicRealm's application of the constructions, in my invalidity analysis with respect to OWS 2.0.

computer system to process other requests." This claim limitation is identical to that of claim 1 with the exception of the "second computer" being used in place of the "Web server" of claim 1. Because the OWS 2.0 HTTP Server (the Web server of claim 1) runs on a second computer (i.e., a computer different from that of the client computer), OWS 2.0 anticipates this claim limitation in the same way it anticipates claim 1.

said page server processing means processing said request and dynamically generating a Web page in response to said request, said Web page including data dynamically retrieved from said one or more data sources.

109. As discussed above with respect to claim 1 of the '554 patent, OWS 2.0 discloses these claim limitations including "intercepting said request at said second computer," "routing said request from said second computer to a dispatcher," and dispatching said request to said page server." These claim limitations are identical to that of claim 1 with the exception of the "second computer" being used in place of the "Web server" of claim 1. Because the OWS 2.0 HTTP Server (the Web server of claim 1) runs on a second computer (i.e., a computer different from that of the client computer), OWS 2.0 anticipates these claim limitations in the same way it anticipates claim 1.

Claim 10 of the '554 patent - The networked system in claim 9 wherein said router in said second computer system includes:

110. As discussed above, OWS 2.0 operates on a network.

an interceptor intercepting said request at said second computer system and routing said request; and

111. As discussed above with respect to claim 1 of the '554 patent, OWS 2.0 discloses this claim limitation including "an interceptor intercepting said request at said second computer system and routing said request." Because OWS 2.0 discloses the step of intercepting a request at a Web server, the OWS HTTP Server (the Web server) includes software for performing that act (i.e., an interceptor).

a dispatcher receiving said routed request from said interceptor and dispatching said request to said page server.

112. As discussed above with respect to claim 1 of the '554 patent, OWS 2.0 discloses this claim limitation including "a dispatcher receiving said routed request from said interceptor and dispatching said request to said page server." Particularly, the OWS 2.0 interceptor of the HTTP Server (the Web server) routes request to the WRB Dispatcher (the dispatcher) for dispatching to a page server.

Claim 11 of the '554 patent

A machine readable medium having stored thereon data representing sequences of instructions, which when executed by a computer system, cause said computer system to perform the steps of:

113. OWS 2.0 is a programmed system whose instructions are on a machine readable medium.[36]

routing a dynamic Web page generation request from a Web server to a page server, said page server receiving said request and releasing said Web server to process other requests wherein said routing step further includes the steps of intercepting said request at said Web server, routing said request from said Web server to a dispatcher, and dispatching said request to said page server;

processing said request, said processing being performed by said page server while said Web server concurrently processes said other requests; and

dynamically generating a Web page, said Web page including data retrieved from one or more data sources.

114. As discussed above with respect to claim 1 of the '554 patent, OWS 2.0 discloses these claim limitations, which are substantially identical to the limitations of claim 1.

Claim 2 of the '335 patent (including limitations of Claim 1)

[36] The parties' proposed constructions for "machine readable medium" are listed in CHART 1 attached to Oracle's Motion for Summary Judgment of Invalidity. I have applied these constructions, and epicRealm's application of the constructions, in my invalidity analysis with respect to OWS 2.0.

A computer-implemented method for managing a dynamic Web page generation request to a Web server, said computer-implemented method comprising the steps of: . . .

115. As discussed above with respect to claim 1 of the '554 patent, OWS 2.0 discloses these claim limitations, which are substantially identical to the limitations of claim 1.

Claim 2 of the 335 patent - The computer-implemented method in claim 1 wherein said step of routing said request includes the steps of: . . .

116. As discussed above with respect to claim 1 of the '554 patent, OWS 2.0 discloses these claim limitations, which are substantially identical to the limitations of claim 1.

Claim 16 of the '335 patent (including limitations of claim 15)

A computer-implemented method comprising the steps of:
transferring a request from an HTTP-compliant device to a page server, . . .

117. As discussed above with respect to claim 1 of the '554 patent, OWS 2.0 discloses these claim limitations, which are substantially identical to the limitations of claim 1. The only significant difference between the limitations of claim 1 of the '554 patent and this claim is that the term "HTTP-compliant device" of this claim is used in place of "Web server" of claim 1. As discussed above, the Oracle WebServer 2.0 is an HTTP-compliant device: "The HTTP Server is the network layer component of the Oracle WebServer. It listens for incoming HTTP requests, delivers static files and runs simple CGI programs and hands off everything else to the Web Request Broker (WRB), where a request is handled by a server extension." A44 (*OWS 2.0 Technical Note)* at ORCL0000764.

Claim 16 of the '335 patent - The computer-implemented method in claim 15 wherein said step of transferring said request includes the steps of: . . .

118. As discussed above with respect to claim 1 of the '554 patent, OWS 2.0 discloses these claim limitations, which are substantially identical to the claim limitations of claim 1.

C. All the Asserted Claims Are Anticipated by Popp

119. As demonstrated below, Popp, including the WebObjects system described therein, anticipates all of the Asserted Claims of the epicRealm patents.

Claim 1 of the '554 patent

A computer-implemented method for managing a dynamic Web page generation request to a Web server, said computer-implemented method comprising the steps of:

120. Popp discloses a computer-implemented method for producing hypertext pages (dynamic web pages) in response to queries (requests). For example, Popp discloses in its Abstract that "some or all of a Web page can be generated dynamically using input received in a returned page, generated at runtime, or retrieved from an external data source (e.g., database or electronic mail system)." A45 (*Popp '291 Patent*) at abstract.

routing said request from said Web server to a page server, said page server receiving said request and releasing said Web server to process other requests,

121. Popp discloses routing a dynamic Web page generation request[37] from a Web server (the HTTP server 206 of Fig. 2 at MAC003930 [A45])[38] to a page server (an Application server 214 of Fig. 2 at MAC0003930 [A45]). Particularly, Popp discloses that the HTTP server receives dynamic Web page requests and returns Web pages in response to those requests: "A client request is forwarded to application server 316 via HTTP server 314." *Id.* at col. 8:20-21.

---

[37] This "request" meets both parties' proposed constructions for that term, namely, "a message that asks for a Web page specified by a URL" as proposed by Oracle and, accordingly, "a message that asks for a Web page" as proposed by epicRealm.

[38] The Popp HTTP server meets both parties' proposed constructions for "Web server," namely, an "HTTP-compliant server software, or a machine running such software, that receives Web page requests and returns Web pages in response to the requests" as proposed by Oracle and, accordingly, "software, or a machine having software, that receives Web page requests and returns Web pages in response to the requests" as proposed by epicRealm.

The routing of Web page requests between the WWW client 202 and the HTTP server 206 and between the HTTP server 206 and an Application 214 is shown in Fig. 2. Id. at MAC003930.

122. The application servers 214 (or just "applications") are the "page servers" of the epicRealm patents under both parties' proposed constructions for that term.[39] Popp describes applications 214 as being able to generate and return "a Web page (e.g., an HTML Web page) that is dynamically generated using complex queries (or other data retrieval mechanisms) to retrieve data and dynamically generate an HTML page using complex logic." Particularly, a Popp application 214 is a "page server" under epicRealm's proposed constructions because it includes software for generating dynamic Web pages. For example, one of the main purposes of application 214 is to receive search requests from a client (i.e., a dynamic Web page request), dynamically generate the results in the form of a Web page, and return the results to the client:

> The application [214] preferably includes an object class hierarchy that can be implemented in any object-oriented language such as objective C, SmallTalk or JAVA. Objects (e.g., Web page definitional objects and control objects) dynamically generate a Web page and manage client interaction via the Internet and WWW.

Id. at col. 7:52-58. Similarly, an application 214 is a "page server" under Oracle's proposed construction for that term because Popp discloses, for example, that "Application 214 can execute on the same or different server as CGIMessenger 210 and/or HTTP Server 206," i.e., it would be possible to run the server on a different machine (i.e., different processor) from that of

[39] The parties' proposed constructions for "page server" are listed in CHART 1 attached to Oracle's Motion for Summary Judgment of Invalidity.

the Web server.[40] *Id.* at col. 7:28-30.

123. Additionally, Popp discloses "releasing said Web Server to process other requests" under both parties' proposed constructions for that claim term.[41] For example, requests are sent from the HTTP server to the application server 214 via CGI. *Id.* at col. 6:49-58. CGI communication is via socket-bind-listen, so SBL releasing is performed, according to epicRealm's SBL releasing theory. See explanation at fn.8 above. Therefore the Web server is released via SBL releasing per epicRealm's application of the proposed "releasing" claim constructions. Additionally, the HTTP server and Application servers 214 can communicate via TCP. Thus under epicRealm's TCP releasing theory with respect to the claimed act of "releasing the Web server," with which I disagree, the HTTP server's receipt of a TCP ACK message from an application server would meet both parties' proposed constructions for "releasing." *Id.*

124. Finally, epicRealm has asserted in this case that the "releasing" of the Web server by the page server can occur simply by virtue of the fact that the page server receives the request and, as a result, implicitly releases (or frees) the Web server to process other requests. Although I disagree with epicRealm's implicit releasing theory, Popp would disclose such releasing in the event the Court adopts epicRealm's theory because, according to epicRealm, the application server's receipt of the request would relieve the HTTP server of the obligation to process it.

> wherein said routing step further includes the steps of intercepting said request at said Web server,

---

[40] Alternatively, it would be obvious to one of ordinary skill in the art to run the application servers (the page servers) on one or more different machines (i.e., different processors) from that of the Web server because such a configuration would spread out the processing of requests across multiple processors and, as a result, would increase the efficiencies of the request processing.

[41] The parties' proposed constructions for "page server . . . releasing said Web server . . ." are listed in CHART 1 attached to Oracle's Motion for Summary Judgment of Invalidity.

125. Popp further discloses the step of intercepting said request at said Web server. Intercepting occurs when the HTTP server diverts the request to a CGI interface program (e.g., CGIMessenger 210) instead of processing the request itself:

> A simple hypertext request (e.g., a request for another form specified in the request) is serviced by HTTP Server 206 by returning the specified form. A simple hypertext request can be, for example, a request for a static HTML page (e.g., a page that contains static information that is defined prior to execution) that does not have any associated logic.
>
> The present invention provides the ability to satisfy more sophisticated requests. For example, the present invention can be used to access a Web page (e.g., an HTML Web page) that is dynamically generated using complex queries (or other data retrieval mechanisms) to retrieve data and dynamically generate an HTML page using complex logic. For more sophisticated requests, the present invention can execute logic such as CGIMessenger 210 and application 214, for example, to process a client request.

A45 at col. 7:37-52. Accordingly, requests that are intended for an application 214 are intercepted at the HTTP server and are routed to the CGIMessenger 210, and other requests (e.g., static requests) can be handled normally by the HTTP server. Thus the HTTP server performs intercepting.[42] Particularly, under epicRealm's proposed "intercepting" construction, the handling of a dynamic Web request is intercepted at the Web server for the purpose of handing it off to the CGIMessenger. To the extent that epicRealm applies Oracle's proposed "intercepting" construction to Oracle's accused products in the same (or a similar) manner that it is applying its own proposed construction, the HTTP server performs intercepting under Oracle's proposed construction. In other words, to the extent that epicRealm interprets Oracle's proposed "intercepting" construction to mean that the processing of certain requests is interrupted at the Web server for the purpose of being handed off, Popp discloses such intercepting.

---

[42] The parties' proposed constructions for "intercepting said request . . ." are listed in CHART 1 attached to Oracle's Motion for Summary Judgment of Invalidity.

routing said request from said Web server to a dispatcher, and dispatching said request to said page server;

126. Popp discloses routing said request from said Web server (the HTTP server 206) to a dispatcher (a CGI interface program). CGI interface program 322 is a "dispatcher" and performs "dispatching" under both parties' proposed constructions for those terms.[43] Examples of CGI messenger programs disclosed in Popp include CGIMessenger 210 and Web Objects server 912. A45 at col. 9:37-44 and col. 27:39-54. Popp describes dispatching as follows:

> To service the client request, CGIMessenger 210 communicates with HTTP server 206 to obtain information regarding the client request. For example, CGIMessenger 210 obtains any information that accompanied the user request such as form and/or application names and/or user input. The form names can include the name of the form that was submitted to initiate the client request as well as a return form.
>
> The application name identifies the application that services the client request. If an application is specified, CGIMessenger 210 transmits the client request and corresponding information transmitted from HTTP Server 206 to application 214. Application 214 can execute on the same or different server as CGIMessenger 210 and/or HTTP Server 206, for example. Application 214 executes an interaction flow to satisfy the user request. Application 214 can access an external data source such as database 224. Database 224 can be resident on the same server as application 214. Alternatively, database 224 can be resident on a separate server (e.g., a separate database server).
>
> • • •
>
> FIG. 5A provides an example of a process flow executed by the Web Objects server to manage applications. A function of the Web Objects server is to dispatch a request to an application that processes the request.
>
> At step 502, the request is received by the Web Objects server. Using the information contained in the request, the Web Objects server identifies the application that will process the request. At step 504, the Web Objects server determines whether the application is executing either on the same or a different server, for example. If the application is not currently loaded, the Web Objects server initiates the application at step 508 and processing continues at step 506. If the application is executing, processing

[43] The parties' proposed constructions for "dispatcher" and "dispatching said request" are listed in CHART 1 attached to Oracle's Motion for Summary Judgment of Invalidity.

continues to step 506.

At step 506, the request is forwarded to the application.

Id. at cols. 7:16-35 and 27:39-55. Accordingly, CGIMessenger 210 or Web Objects server 912 examines the incoming request and dispatches it based on dynamic information maintained about the applications 214 (the page servers) (e.g., the application name information indicating which application should be used to more efficiently process the request and/or information indicating whether the application is executing on the same or different server). Accordingly, CGI interface program 322 dispatches requests according to epicRealm's proposed construction for that term because it examines requests to make an informed selection of which page server (application server 214) should process requests based on dynamic information maintained about page servers, which is information indicating which page server can more efficiently process the request. Because Oracle's proposed construction for "dispatching" is broader than that of epicRealm, the CGI interface program must also perform dispatching under Oracle's proposed construction.

processing said request, said processing being performed by said page server while said Web server concurrently processes said other requests; and

127. Popp discloses this claim limitation as well. Concurrent processing between the Web server and the page server of Popp are disclosed because Popp teaches that the HTTP server and the application servers can run on different machines (i.e., different processors). *Id.* at col. 7:25-35.

dynamically generating a Web page in response to said request, said Web page including data dynamically retrieved from one or more data sources.

128. Popp discloses this final limitation as well. As discussed above, Popp discloses

dynamically generating a Web page[44] in response to the request which includes data dynamically retrieved from one or more data sources. Particularly, an application 214 returns "a Web page (e.g., an HTML Web page) that is dynamically generated using complex queries (or other data retrieval mechanisms) to retrieve data and dynamically generate an HTML page using complex logic." A45 at col. 7:44-58. Additionally, "[t]he contents of the generated Web page can contain data retrieved from an external data source (e.g., such as database server 318), or state information maintained by the application." *Id.* at col. 8:44-46.

Claim 2 of the '554 Patent - The computer-implemented method in claim 1 wherein said step of processing said request includes the step of identifying said one or more data sources from which to retrieve said data.

129. Popp discloses the step of identifying one or more data sources[45] from which to retrieve said data. For example, Popp discloses that a CGI program (e.g., CGI server 906) can identify a Web Object server 912 based on the information contained in the request, which can include a data source from which to retrieve data. *Id.* at col. 26:3-4.

Claim 3 of the '554 Patent - The computer-implemented method in claim 2 wherein said step of dynamically generating said Web page includes the step of dynamically retrieving said data from said one or more data sources.

130. Popp discloses dynamically retrieving said data from one or more data sources. As discussed above with respect to claim 1, a CGIMessenger 210 can dispatch requests to an application server 214 that can dynamically retrieve data from one or more data sources to

---

[44] The Web page generated by the Popp page server meets both parties' proposed constructions for "Web page," namely, an "a file containing embedded commands in a Web formatting language such as HTML, capable of being displayed on a Web browser" as proposed by Oracle and, accordingly, "Web content displayable through a Web browser" as proposed by epicRealm.

[45] The parties' proposed constructions for "data sources" are listed in CHART 1 attached to Oracle's Motion for Summary Judgment of Invalidity. I have applied these constructions, and epicRealm's application of the constructions, in my invalidity analysis with respect to Popp.

include in the dynamic Web page: "the present invention can be used to access a Web page (e.g., an HTML Web page) that is dynamically generated using complex queries (or other data retrieval mechanisms) to retrieve data [from data sources] and dynamically generate an HTML page using complex logic." A45 at col. 7:45-49.

Claim 4 of the '554 Patent - The computer-implemented method in claim 3 wherein said step of processing said request includes the step of said page server maintaining a connection cache to said one or more data sources.

131. Popp discloses maintaining a connection cache to said one or more data sources.[46] Particularly, Popp discloses: "State information can be retained to allow the internal application to allow the application to maintain open sessions with multiple users capable of processing multiple transactions in any order." *Id.* at col. 8:28-31. Further, "a connection to a database can be used at the global level by multiple sessions and applications." *Id.* at col. 13:38-40.

Claim 5 of the '554 patent - The computer-implemented method in claim 3 wherein said step of processing said request includes the step of logging into said one or more data sources.

132. Popp discloses logging into[47] said one or more data sources. For example, Popp discloses: "A digital signature can be used as a key to perform verification. For example, the digital signature can be used to verify that a request is a valid request generated by a valid client. The digital signature can further be used to verify that the client is authorized to access the data of the specified session." A45 at col. 27:12-17.

Claim 7 of the '554 patent - The computer-implemented method in claim 3 wherein said page server includes custom HTML extension templates for configuring said Web page.

---

[46] The parties' proposed constructions for "connection cache" are listed in CHART 1 attached to Oracle's Motion for Summary Judgment of Invalidity. I have applied these constructions, and epicRealm's application of the constructions, in my invalidity analysis with respect to Popp.

[47] The parties' proposed constructions for "logging into" are listed in CHART 1 attached to Oracle's Motion for Summary Judgment of Invalidity. I have applied these constructions, and epicRealm's application of the constructions, in my invalidity analysis with respect to Popp.

133. Popp discloses a page server that includes custom HTML extension templates for configuring a Web page. For example, Popp discloses: "The present invention provides an extension to HTML that is used on the HTTP server side. The HTML extension is filtered out before a Web page is sent to a client browser. It is used to interpret an HTML template and to render an HTML document before is transmitted to the client browser." *Id.* at col. 15:45-50.

Claim 8 of the '554 patent - The computer-implemented method in claim 7 wherein said step of processing said request further includes the step of inserting said dynamically retrieved data from said one or more data sources into said custom HTML extension templates.

134. Popp discloses inserting said dynamically retrieved data from said one or more data sources into said custom HTML extension templates. For example, Popp discloses: "A stringcontrol object instance instantiates an NSWHString static HTML element and inserts it in the template." *Id.* at col. 18:57-59.

Claim 9 of the '554 patent

Claim 9 of the '554 patent - A networked system for managing a dynamic Web page generation request, said system comprising:

135. Popp teaches a networked system for managing dynamic page generation requests: "Some or all of a Web page can be generated dynamically using input received in a returned page, generated at runtime, or retrieved from an external data source (e.g., database or electronic mail system). *Id.* at Abstract. Additionally, a network is disclosed as follows: "[t]he present invention provides the ability to access an application using any interface. For example, a client can access the same application via the Internet using a system running Windows, MAC/OS, Sun.OS, NextStep, etc. Referring to FIG. 2, client 202 is resident on a corporate network 200. Corporate network 200 is a local area network comprised of personal computers such as client 202, for example." *Id.* at col. 6:32-40.

one or more data sources;

136. As discussed above with respect to claim 1 of the '554 patent, Popp discloses a networked system with one or more data sources. Particularly, "[t]he contents of the Web page can contain data retrieved from an external data source (e.g., such as database server 318)." A45 at col. 8:44-46.

a page server having a processing means;

137. As discussed above with respect to claim 1, Popp discloses a page server having a processing means. Particularly, as discussed above, application servers 214 are page servers. The processors on the computer(s) that run these applications 214 are the "page server processing means" under both parties' proposed constructions for that term.[48]

a first computer system including means for generating said request; and

138. Popp discloses a first computer system including means for generating said request.[49] The first computer system of Popp is a client computer that has a processor that runs a web browser: "Requests submitted by clients 302, 304, 306, and 308 to HTTP server 314 are transmitted to HTTP server 314 via WWW 310." *Id.* at col. 8:17-19.

a second computer system including means for receiving said request from said first computer,

139. Popp discloses a second computer system including means for receiving said

[48] The parties' proposed constructions for "page server processing means" are listed in CHART 1 attached to Oracle's Motion for Summary Judgment of Invalidity. I have applied these constructions, and epicRealm's application of the constructions, in my invalidity analysis with respect to Popp.

[49] The parties' proposed constructions for "means for generating said request" are listed in CHART 1 attached to Oracle's Motion for Summary Judgment of Invalidity. I have applied these constructions, and epicRealm's application of the constructions, in my invalidity analysis with respect to Popp.

request from said first computer.[50] The second computer system is box 208 of Figure 2 including, for example, the HTTP Server 206 and the CGIMessenger 210.

said second computer system also including a router, said router routing said request from said second computer system to said page server,

140. As discussed above with respect to claim 1 of the '554 patent, Popp discloses a second computer system including means for receiving said request from said first computer. That second computer system includes a router[51] (including the HTTP server and the CGIMessenger) which, as discussed above, performs the functions of intercepting requests at the Web server and dispatching them to page servers (application servers 214).

wherein said routing further includes intercepting said request at said second computer, routing said request from said second computer to a dispatcher, and dispatching said request to said page server

141. As discussed above with respect to claim 1 of the '554 patent, Popp discloses these claim limitations including "intercepting said request at said second computer," "routing said request from said second computer to a dispatcher," and dispatching said request to said page server." These claim limitations are identical to that of claim 1 with the exception of the "second computer" being used in place of the "Web server" of claim 1. Because the HTTP server 206 (the Web server of claim 1) runs on a second computer (i.e., a computer different from that of the client computer), Popp anticipates these claim limitations in the same way it anticipates claim 1.

---

[50] The parties' proposed constructions for "means for receiving said request" are listed in CHART 1 attached to Oracle's Motion for Summary Judgment of Invalidity. I have applied these constructions, and epicRealm's application of the constructions, in my invalidity analysis with respect to Popp.

[51] The parties' proposed constructions for "router" are listed in CHART 1 attached to Oracle's Motion for Summary Judgment of Invalidity. I have applied these constructions, and epicRealm's application of the constructions, in my invalidity analysis with respect to Popp.

said page server receiving said request and releasing said second computer system to process other requests,

142. As discussed above with respect to claim 1 of the '554 patent, Popp discloses this claim limitation including "said page server receiving said request and releasing said second computer system to process other requests." This claim limitation is identical to that of claim 1 with the exception of the "second computer" being used in place of the "Web server" of claim 1. Because the HTTP server 206 (the Web server of claim 1) runs on a second computer (i.e., a computer different from that of the client computer), Popp anticipates this claim limitation in the same way it anticipates claim 1.

said page server processing means processing said request and dynamically generating a Web page in response to said request, said Web page including data dynamically retrieved from said one or more data sources.

143. As discussed above with respect to claim 1 of the '554 patent, Popp discloses these claim limitations including "intercepting said request at said second computer," "routing said request from said second computer to a dispatcher," and dispatching said request to said page server." These claim limitations are identical to that of claim 1 with the exception of the "second computer" being used in place of the "Web server" of claim 1. Because the HTTP server 206 (the Web server of claim 1) runs on a second computer (i.e., a computer different from that of the client computer), Popp anticipates these claim limitations in the same way it anticipates claim 1.

Claim 10 of the '554 patent - The networked system in claim 9 wherein said router in said second computer system includes:

144. As discussed above, Popp discloses a networked system including a second computer system.

an interceptor intercepting said request at said second computer system and routing said request; and

145. As discussed above with respect to claim 1 of the '554 patent, Popp discloses this

claim limitation including "an interceptor intercepting said request at said second computer system and routing said request." Because Popp discloses the step of intercepting a request at a Web server, the HTTP server (the Web server) includes software for performing that act (i.e., an interceptor).

a dispatcher receiving said routed request from said interceptor and dispatching said request to said page server.

146. As discussed above with respect to claim 1 of the '554 patent, Popp discloses this claim limitation including "a dispatcher receiving said routed request from said interceptor and dispatching said request to said page server." Particularly, the interceptor of the HTTP server (the Web server) routes request to the CGIMessenger (the dispatcher) for dispatching to a page server.

Claim 11 of the '554 patent

A machine readable medium having stored thereon data representing sequences of instructions, which when executed by a computer system, cause said computer system to perform the steps of: .. .

147. The Popp server system is a programmed system whose instructions are on a machine readable medium.[52]

148. As discussed above with respect to claim 1 of the '554 patent, Popp also discloses this claim's additional limitations, which are substantially identical to the limitations of claim 1.

Claim 2 of the '335 patent (including limitations of Claim 1)

A computer-implemented method for managing a dynamic Web page generation request to a Web server, said computer-implemented method comprising the steps of: . . .

[52] The parties' proposed constructions for "machine readable medium" are listed in CHART 1 attached to Oracle's Motion for Summary Judgment of Invalidity. I have applied these constructions, and epicRealm's application of the constructions, in my invalidity analysis with respect to Popp.

149. As discussed above with respect to claim 1 of the '554 patent, Popp discloses these claim limitations, which are substantially identical to the limitations of claim 1.

Claim 2 of the 335 patent - The computer-implemented method in claim 1 wherein said step of routing said request includes the steps of: .

150. As discussed above with respect to claim 1 of the '554 patent, Popp discloses these claim limitations, which are substantially identical to the limitations of claim 1.

Claim 16 of the '335 patent (including limitations of claim 15)

A computer-implemented method comprising the steps of: transferring a request from an HTTP-compliant device to a page server, .

151. As discussed above with respect to claim 1 of the '554 patent, Popp discloses these claim limitations, which are substantially identical to the limitations of claim 1. The only significant difference between the limitations of claim 1 of the '554 patent and this claim is that the term "HTTP-compliant device" of this claim is used in place of "Web server" of claim 1. As discussed above, the HTTP server of the Popp server system is an HTTP-compliant device hence its name "HTTP server."

Claim 16 of the '335 patent - The computer-implemented method in claim 15 wherein said step of transferring said request includes the steps of: .. .

152. As discussed above with respect to claim 1 of the '554 patent, Popp discloses these claim limitations, which are substantially identical to the claim limitations of claim 1.

D. All the Asserted Claims Are Anticipated by or Obvious in View of Dienst

153. As demonstrated below, the Dienst reference, including the system described therein, anticipates all of the Asserted Claims of the epicRealm patents.

Claim 1 of the '554 patent

A computer-implemented method for managing a dynamic Web page generation request to a Web server, said computer-implemented method comprising the steps of:

154. Dienst discloses a computer-implemented method for producing hypertext pages

(dynamic web pages) in response to queries (requests). For example, Dienst discloses: "The results of a search (that may have been dispatched to several servers) are presented to the user as a combined list of hyperlinks sorted by publisher and docid." A46 (Dienst) at MAC000281. These results that are presented to the user are dynamic Web pages.

routing said request from said Web server to a page server, said page server receiving said request and releasing said Web server to process other requests,

155. Dienst discloses routing a dynamic Web page generation request[53] from a Web server (the WWW server of Fig. 2 at MAC000279 [A461)[54] to a page server (a Dienst server of Fig. 2 at MAC000279[A461). Particularly, Dienst describes the WWW server as an HTTP server for receiving Web page requests and returning Web pages in response to those requests:

> World Wide Web server - The system is designed for access from Web clients through a Web server. Dienst may be accessed through any Web server that supports the Common Gateway Interface (CGI). The current version of Dienst has been tested with the two most popular servers, CERN and NCSA.
>
> Dienst requests are packaged within the path portion of the HTTP request to the Web server. A Web server that serves as a front-end for Dienst must be configured to invoke the Dienst CGI stub program when the HTTP request contains a Dienst request. Setting up a Web server as a gateway to Dienst server is described in section 6.

Id. at MAC000280. The routing of Web page requests between the WWW client and the WWW server and between the WWW server and a Dienst server is shown in Fig. 2.

---

[53] This "request" meets both parties' proposed constructions for that term, namely, "a message that asks for a Web page specified by a URL" as proposed by Oracle and, accordingly, "a message that asks for a Web page" as proposed by epicRealm.

[54] The Dienst WWW server meets both parties' proposed constructions for "Web server," namely, an "HTTP-compliant server software, or a machine running such software, that receives Web page requests and returns Web pages in response to the requests" as proposed by Oracle and, accordingly, "software, or a machine having software, that receives Web page requests and returns Web pages in response to the requests" as proposed by epicRealm.

156. The Dienst servers are the "page servers" of the epicRealm patents under both parties' proposed constructions for that term.[55] Dienst describes Dienst servers as follows:

> Dienst server - A Dienst server is a standalone multi-threaded process written in Perl. The server listens for connections on a port defined in the server configuration file. Valid requests to the server are defined by the Dienst protocol. All responses from Dienst are formatted in the same fashion as HTTP responses. These responses are sent to the requesting client without interpretation by the Web server or CGI stub program. As part of fulfilling a request, an individual Dienst server may make requests to and interpret responses from other Dienst servers. Interoperation among servers is described in section 4.2.

Id. at MAC000280. Particularly, a Dienst server is a "page server" under epicRealm's proposed constructions because it includes software for generating dynamic Web pages. For example, one of the main purposes of the Dienst server is to receive search requests from a client (i.e., a dynamic Web page request), dynamically generate the results in the form of a "hit list," and return the results to the client:

> Each Dienst user interface server provides a front-end for searching the collection provided by all inter-operating Dienst servers. The user specifies in the search form which publishers should be searched. The respective user interface server then dispatches search requests, in parallel, to the Dienst index servers that correspond to the publisher selection(s). The user interface server then parses and combines the results from the requested servers to present a coordinated "hit list" to the user.

Id. at MAC000281. An example of such a "hit list" (a dynamically generated Web page) is shown in Figure 4 at MAC000283 [A46]. This "hit list" is described as a "unified hypertext hit list," which includes "[t]he results of a search (that may have been dispatched to several servers) are presented to the user as a combined list of hyperlinks sorted by publisher and docid." *Id.* at MAC000281. Similarly, a Dienst server is a "page server" under Oracle's proposed construction

[55] The parties' proposed constructions for "page server" are listed in CHART 1 attached to Oracle's Motion for Summary Judgment of Invalidity.

for that term because Dienst suggests, for example, that it would be possible to run the server on a different machine (i.e., different processor) from that of the Web server (the WWW server) over a "distributed network."[56] See, e.g., id. at Figure 2 at MAC000279; MAC000310.

157. Additionally, Dienst discloses "releasing said Web Server to process other requests" under both parties' proposed constructions for that claim term.[57] For example, requests are sent "via socket I/O to the Dienst server." *Id.* at MAC000280. Therefore the Web server is released via SBL releasing per epicRealm's explicit releasing theory. See explanation at fn.8 above. Additionally, "[s]ince Dienst servers are accessed through gateways from Web servers, the Dienst protocol uses HTTP (the protocol of the World Wide Web) as a transport layer for requests." *Id.* at MAC000326. Thus under epicRealm's TCP releasing theory with respect to the claimed act of "releasing the Web server," with which I disagree, the WWW server's receipt of a TCP ACK message from a Dienst server would meet both parties' proposed constructions for "releasing." See *id.*

158. Finally, epicRealm has asserted in this case that the "releasing" of the Web server by the page server can occur simply by virtue of the fact that the page server receives the request and, as a result, implicitly releases (or frees) the Web server to process other requests. Although I disagree with epicRealm's implicit releasing theory, Dienst would disclose such releasing in the event the Court adopts epicRealm's theory because, according to epicRealm, the Dienst server's

---

[56] Alternatively, it would be obvious to one of ordinary skill in the art to run the Dienst servers (the page servers) on one or more different machines (i.e., different processors) from that of the Web server because such a configuration would spread out the processing of requests across multiple processors and, as a result, would increase the efficiencies of the request processing.

[57] The parties' proposed constructions for "page server . . . releasing said Web server . . ." are listed in CHART 1 attached to Oracle's Motion for Summary Judgment of Invalidity.

receipt of the request would relieve the WWW server of the obligation to process it.

wherein said routing step further includes the steps of intercepting said request at said Web server,

159. Dienst further discloses the step of intercepting said request at said Web server. Interception occurs when the WWW server recognizes a request as a "Dienst request" that must be processed by a Dienst server: "Dienst requests are packaged within the path portion of the HTTP request to the Web server. A Web server that serves as a front-end for Dienst must be configured to invoke the Dienst CGI stub program when the HTTP request contains a Dienst request." A46 at MAC000280 (further disclosing that setting up a Web server to perform intercepting is described in section 6 of Dienst). Accordingly, requests that are intended for the Dienst server are intercepted at the WWW server and are routed to the CGI stub for handling, and all other requests are handled normally by the WWW server. Thus the Dienst WWW server performs intercepting.[58] Particularly, under epicRealm's proposed "intercepting" construction, the handling of a dynamic Web page request is intercepted at the Web server for the purpose of handing it off to the CGI stub. To the extent that epicRealm applies Oracle's proposed "intercepting" construction to Oracle's accused products in the same (or a similar) manner that it is applying its own proposed construction, the WWW server performs intercepting under Oracle's proposed construction. In other words, to the extent that epicRealm interprets Oracle's proposed "intercepting" construction to mean that the processing of certain requests is interrupted at the Web server for the purpose of being handed off, Dienst discloses such intercepting.

[58] The parties' proposed constructions for "intercepting said request . . ." are listed in CHART 1 attached to Oracle's Motion for Summary Judgment of Invalidity.

routing said request from said Web server to a dispatcher, and dispatching said request to said page server;

160. Dienst discloses routing said request from said Web server (the WWW server) to a dispatcher. The CGI stub (including, in some cases, the portion of a Dienst server which recognizes the need to request content from a different Dienst server) is a "dispatcher" and performs "dispatching" under both parties' proposed constructions for those terms.[59] Dienst describes the Dienst CGI stub as follows:

> Dienst CGI stub - As described above, the CGI stub is invoked each time the gateway Web server receives a Dienst request. The stub is a small, simple executable that strips the Dienst request from the HTTP request, packages a few important environment variables that contain information about the request . . . and sends them via socket I/O to the Dienst server.

A46 at MAC000280. Additionally, Dienst discloses: "As part of fulfilling a request, an individual Dienst server may make requests to and interpret responses from other Dienst servers. Interoperation among servers is described in section 4.2." Id. at MAC000280.

161. As disclosed in section 4.2 of Dienst, after the CGI stub sends a request to the interfacing Dienst server, that server can further "dispatch" requests to other Dienst servers ("Dienst index servers") that correspond to the publisher selections identified in the search request. Id. at MAC000281. Once the search request is examined and the selected publishers are identified, requests are dispatched based on dynamic information maintained about other Dienst servers (page servers). Particularly, a "meta-server" maintains dynamic information about the other Dienst servers:

> Each server in a set of interoperating Dienst servers indexes and acts as a repository for documents from one or more publishers. In

[59] The parties' proposed constructions for "dispatcher" and "dispatching said request" are listed in CHART 1 attached to Oracle's Motion for Summary Judgment of Invalidity.

> the current version, all the documents for a publisher must be indexed and reside on the same server. Each server is able to locate the indexing and repository site for a specific publisher using tables that are maintained by a distinguished server, the meta server, and periodically downloaded by the individual server.

Id. at MAC000281. Accordingly, the CGI stub and interfacing Dienst server dispatch requests according to epicRealm's proposed construction for that term because they examine requests to make an informed selection of which page server (Dienst server) should process requests based on dynamic information maintained about page servers, which is information indicating which page server can more efficiently process the request. Because Oracle's proposed construction for "dispatching" is broader than that of epicRealm, the CGI stub and interfacing Dienst server must also perform dispatching under Oracle's proposed construction.

processing said request, said processing being performed by said page server while said Web server concurrently processes said other requests; and

162. Dienst discloses this claim limitation as well. Concurrency is inherent in the architecture of Figure 2. A46 at MAC000279. When the WWW server has passed on the request to a Dienst server, it is free to process other requests concurrently. Also, Dienst discloses the ability of running other Dienst servers and "foreign servers" over a distributed network, which would run on separate machines from the Web server. See *Id.* at MAC000304 and 310.

dynamically generating a Web page in response to said request, said Web page including data dynamically retrieved from one or more data sources.

163. Dienst discloses this final limitation as well. Dienst discloses dynamically generating a Web page[60] in response to the request which includes data dynamically retrieved

[60] The Web page generated by the Dienst page server meets both parties' proposed constructions for "Web page," namely, an "a file containing embedded commands in a Web formatting language such as HTML, capable of being displayed on a Web browser" as proposed by Oracle and, accordingly, "Web content displayable through a Web browser" as proposed by epicRealm.

from one or more data sources (e.g., Dienst servers other than the server interfacing with the WWW server). A Dienst server's generation of a dynamic Web page is described as follows: "The user interface server [a Dienst server] then parses and combines the results from the requested servers to present a coordinated 'hit list' to the user." A46 at MAC000281-283. This hypertext "hit list" (an example of which is shown at MAC000284) is a Web page under both parties' construction for that term. Additionally, depending on the request, the generated Web page can include data (e.g., search results) dynamically retrieved from one or more other Dienst servers (the data sources): "A search request is not limited to the server to which the client is connected. The client may choose to submit the search in parallel to multiple Dienst servers and view the combined results." *Id.* at MAC000275.

Claim 2 of the '554 Patent - The computer-implemented method in claim 1 wherein said step of processing said request includes the step of identifying said one or more data sources from which to retrieve said data.

164. Dienst discloses the step of identifying one or more data sources[61] from which to retrieve said data. The data source is identified in the HTTP request: "The stub is a small, simple executable that strips the Dienst request from the HTTP request, packages a few important environment variables that contain information about the request (e.g., the remote host identity, the MIME types the client is willing to accept, etc.), and sends them via socket I/O to the Dienst server." A46 at MAC000280. Dienst must also identify data sources to invoke "foreign servers." *Id.* at MAC000304 and 310.

Claim 3 of the '554 Patent - The computer-implemented method in claim 2 wherein said step of dynamically generating said Web page includes the step of dynamically retrieving said data from said one or more data sources.

[61] The parties' proposed constructions for "data sources" are listed in CHART 1 attached to Oracle's Motion for Summary Judgment of Invalidity. I have applied these constructions, and epicRealm's application of the constructions, in my invalidity analysis with respect to Dienst.

165. Dienst discloses dynamically retrieving said data from one or more data sources. As discussed above with respect to claim 1, an interface Dienst server can dispatch requests to one or more other Dienst servers and dynamically retrieve data from those sources to include in the dynamic Web page: "The respective user interface server then dispatches search requests, in parallel, to the Dienst index servers that correspond to the publisher selection(s). The user interface server then parses and combines the results from the requested servers to present a coordinated 'hit list' to the user." *Id.* at MAC000281.

Claim 4 of the '554 Patent - The computer-implemented method in claim 3 wherein said step of processing said request includes the step of said page server maintaining a connection cache to said one or more data sources.

166. It would be obvious to one of ordinary skill in the art to combine the step of maintaining a connection cache to said one or more data sources as disclosed in OWS 1.0, OWS 2.0, and Popp with Dienst. One of ordinary skill in the art would be motivated to combine these prior art references because they are all are directed to the management of dynamic Web page generation requests and the use of a connection cache eliminates subsequent connect times to data sources.

Claim 5 of the '554 patent - The computer-implemented method in claim 3 wherein said step of processing said request includes the step of logging into said one or more data sources.

167. It would be obvious to one of ordinary skill in the art to combine the step of logging into said one or more data sources as disclosed in OWS 1.0, OWS 2.0, and Popp with Dienst. One of ordinary skill in the art would be motivated to combine these prior art references because they are all are directed to the management of dynamic Web page generation requests and it may be desired to require the presentation of credentials to access the one or more data sources.

Claim 7 of the '554 patent - The computer-implemented method in claim 3 wherein said page server includes custom HTML extension templates for configuring said Web page.

168. It would be obvious to one of ordinary skill in the art to combine the use of custom HTML extension templates as disclosed in OWS 1.0, OWS 2.0, and Popp with Dienst. One of ordinary skill in the art would be motivated to combine these prior art references because they are all are directed to the management of dynamic Web page generation requests and it may be desired to use such extension templates for configuring Web pages.

Claim 8 of the '554 patent - The computer-implemented method in claim 7 wherein said step of processing said request further includes the step of inserting said dynamically retrieved data from said one or more data sources into said custom HTML extension templates.

169. It would be obvious to one of ordinary skill in the art to combine the use of custom HTML extension templates as disclosed in OWS 1.0, OWS 2.0, and Popp with Dienst. One of ordinary skill in the art would be motivated to combine these prior art references because they are all are directed to the management of dynamic Web page generation requests and it may be desired to use such extension templates for configuring Web pages.

Claim 9 of the '554 patent

Claim 9 of the '554 patent - A networked system for managing a dynamic Web page generation request, said system comprising:

170. Dienst teaches a networked system for managing dynamic page generation requests: "The basic Dienst server interoperates with a distributed network of other Dienst servers." A46 at MAC000310.

one or more data sources;

171. As discussed above with respect to claim 1 of the '554 patent, Dienst discloses a networked system with one or more data sources. Particularly, Dienst is replete with references to "other Dienst servers" that can supply data (e.g., search results) to an interfacing Dienst server for the purpose of generating a Web page. *Id.* at MAC000275.

a page server having a processing means;

172. As discussed above with respect to claim 1, Dienst discloses a page server having a processing means. Particularly, as discussed above, Dienst servers are page servers. The processors on the computer(s) that run these Dienst servers are the "page server processing means" under both parties' proposed constructions for that term.[62]

a first computer system including means for generating said request; and

173. Dienst discloses a first computer system including means for generating said request.[63] The first computer system of Dienst is a client computer that has a processor that runs a web browser. See, e.g., the "WWW Client" computer of Figure 2. *Id.* at MAC000279.

a second computer system including means for receiving said request from said first computer,

174. Dienst discloses a second computer system including means for receiving said request from said first computer.[64] The second computer system is the system running the WWW server, CGI stub and Dienst server. See, e.g., the computer system of Figure 2 at *Id.* at MAC000279.

said second computer system also including a router, said router routing said request from said second computer system to said page server,

---

[62] The parties' proposed constructions for "page server processing means" are listed in CHART 1 attached to Oracle's Motion for Summary Judgment of Invalidity. I have applied these constructions, and epicRealm's application of the constructions, in my invalidity analysis with respect to Dienst.

[63] The parties' proposed constructions for "means for generating said request" are listed in CHART 1 attached to Oracle's Motion for Summary Judgment of Invalidity. I have applied these constructions, and epicRealm's application of the constructions, in my invalidity analysis with respect to Dienst.

[64] The parties' proposed constructions for "means for receiving said request" are listed in CHART 1 attached to Oracle's Motion for Summary Judgment of Invalidity. I have applied these constructions, and epicRealm's application of the constructions, in my invalidity analysis with respect to Dienst.

175. As discussed above with respect to claim 1 of the '554 patent, Dienst discloses a second computer system including means for receiving said request from said first computer. That second computer system includes a router[65] (including the WWW server and CGI stub) which, as discussed above, performs the functions of intercepting requests at the Web server and dispatching them to page servers (Dienst servers).

wherein said routing further includes intercepting said request at said second computer, routing said request from said second computer to a dispatcher, and dispatching said request to said page server

176. As discussed above with respect to claim 1 of the '554 patent, Dienst discloses these claim limitations including "intercepting said request at said second computer," "routing said request from said second computer to a dispatcher," and dispatching said request to said page server." These claim limitations are identical to that of claim 1 with the exception of the "second computer" being used in place of the "Web server" of claim 1. Because the WWW server (the Web server of claim 1) runs on a second computer (i.e., a computer different from that of the client computer), Dienst anticipates these claim limitations in the same way it anticipates claim 1.

said page server receiving said request and releasing said second computer system to process other requests,

177. As discussed above with respect to claim 1 of the '554 patent, Dienst discloses this claim limitation including "said page server receiving said request and releasing said second computer system to process other requests." This claim limitation is identical to that of claim 1 with the exception of the "second computer" being used in place of the "Web server" of claim 1.

---

[65] The parties' proposed constructions for "router" are listed in CHART 1 attached to Oracle's Motion for Summary Judgment of Invalidity. I have applied these constructions, and epicRealm's application of the constructions, in my invalidity analysis with respect to Dienst.

Because the WWW server (the Web server of claim 1) runs on a second computer (i.e., a computer different from that of the client computer), Dienst anticipates this claim limitation in the same way it anticipates claim 1.

said page server processing means processing said request and dynamically generating a Web page in response to said request, said Web page including data dynamically retrieved from said one or more data sources.

178. As discussed above with respect to claim 1 of the '554 patent, Dienst discloses these claim limitations including "intercepting said request at said second computer," "routing said request from said second computer to a dispatcher," and dispatching said request to said page server." These claim limitations are identical to that of claim 1 with the exception of the "second computer" being used in place of the "Web server" of claim 1. Because the WWW server (the Web server of claim 1) runs on a second computer (i.e., a computer different from that of the client computer), Dienst anticipates these claim limitations in the same way it anticipates claim 1.

Claim 10 of the '554 patent - The networked system in claim 9 wherein said router in said second computer system includes:

179. As discussed above, Dienst discloses a networked system including a second computer system.

an interceptor intercepting said request at said second computer system and routing said request; and

180. As discussed above with respect to claim 1 of the '554 patent, Dienst discloses this claim limitation including "an interceptor intercepting said request at said second computer system and routing said request." Because Dienst discloses the step of intercepting a request at a Web server, the WWW server (the Web server) includes software for performing that act (i.e., an interceptor).

a dispatcher receiving said routed request from said interceptor and dispatching said request to said page server.

181. As discussed above with respect to claim 1 of the '554 patent, Dienst discloses this claim limitation including "a dispatcher receiving said routed request from said interceptor and dispatching said request to said page server." Particularly, the interceptor of the WWW server (the Web server) routes request to the CGI stub (the dispatcher) for dispatching to a page server.

Claim 11 of the '554 patent

A machine readable medium having stored thereon data representing sequences of instructions, which when executed by a computer system, cause said computer system to perform the steps of: .. .

182. The Dienst server system is a programmed system whose instructions are on a machine readable medium.[66]

183. As discussed above with respect to claim 1 of the '554 patent, Dienst discloses this claim's additional limitations, which are substantially identical to the limitations of claim 1.

Claim 2 of the '335 patent (including limitations of Claim 1)

A computer-implemented method for managing a dynamic Web page generation request to a Web server, said computer-implemented method comprising the steps of: .

184. As discussed above with respect to claim 1 of the '554 patent, Dienst discloses these claim limitations, which are substantially identical to the limitations of claim 1.

Claim 2 of the 335 patent - The computer-implemented method in claim 1 wherein said step of routing said request includes the steps of: .

185. As discussed above with respect to claim 1 of the '554 patent, Dienst discloses these claim limitations, which are substantially identical to the limitations of claim 1.

---

[66] The parties' proposed constructions for "machine readable medium" are listed in CHART 1 attached to Oracle's Motion for Summary Judgment of Invalidity. I have applied these constructions, and epicRealm's application of the constructions, in my invalidity analysis with respect to Dienst.

Claim 16 of the '335 patent (including limitations of claim 15)

A computer-implemented method comprising the steps of:
transferring a request from an HTTP-compliant device to a page server, .

186. As discussed above with respect to claim 1 of the '554 patent, Dienst discloses these claim limitations, which are substantially identical to the limitations of claim 1. The only significant difference between the limitations of claim 1 of the '554 patent and this claim is that the term "HTTP-compliant device" of this claim is used in place of "Web server" of claim 1. As discussed above, the WWW server of the Dienst server system is an HTTP-compliant device: "A Web server that serves as a front-end for Dienst must be configured to invoke the Dienst CGI stub program when the HTTP request contains a Dienst request." A46 at MAC000280.

Claim 16 of the '335 patent - The computer-implemented method in claim 15 wherein said step of transferring said request includes the steps of: .. .

187. As discussed above with respect to claim 1 of the '554 patent, Dienst discloses these claim limitations, which are substantially identical to the claim limitations of claim 1.

E. All the Asserted Claims Are Obvious in View of OWS 1.0, OWS 2.0, Dienst or Popp in Combination with Garland

188. Another reference that discloses dispatching of Web page requests according to the epicRealm patents is Garland et al., "Implementing Distributed Server Groups for the World Wide Web," Carnegie Mellon University Technical Report CMUCS-95-114 (Jan. 1995) ("Garland"). Appendix at A48. Like many other prior art references from 1995 and earlier, Garland identified the problem of web servers handling too much load on a single server machine:

> As the exponential growth of the World Wide Web continues with no near-term end in sight, the load placed on popular Web servers is often too great for a single server machine to handle. When the volume of requests to a server becomes prohibitively high the server tends to drop requests and eventually crashes. The service provider needs more computing power to meet the demands of the clients requesting the service. One solution is to simply buy a bigger machine that can handle the large load of service requests.

> This solution is, however, unacceptable; it provides only temporary relief from server overload. If server usage continues to grow rapidly, even the more powerful server will eventually be overwhelmed. This problem is likely to become more significant over the next few years as the number of commercial service providers and the size of the data objects being exchanged are likely to increase dramatically. *Id.* at Introduction.

Garland also identified that the "obvious" solution to the above problem was to implement a distributed Web server architecture and dynamic load balancing techniques to manage incoming HTTP (page) requests by distributing them among server machines using a dispatcher server:

> A fairly obvious solution to this problem is to implement a distributed Web server which will spread incoming request load among several machines. . . . By supporting distributed Web server groups, we can provide a mechanism for smoothly scaling the server capacity of Web service providers. The use of server groups can reduce the latency of servicing a request provided there is sufficient available network bandwidth to accommodate higher request throughput. Our Web server group implementation can effectively balance request load, and is transparent to Web clients which support the HTTP/1.0 protocol. *Id.* at Introduction.

Garland further describes a particular distributed web server architecture where a "dispatcher server" receives web page requests and dispatches them to appropriate member servers such that the dispatcher server and the several member servers can concurrently process requests:

> We have chosen the following general architecture for our Web server groups: • There is a special server (the dispatcher) which coordinates and controls the server group. Requests intended for the server group are received by the dispatcher. • Attached to the dispatcher is a collection of member servers. These servers are the actual repository of data and fulfill requests made of the group. The dispatcher is the single entity representing the group to the outside world. Incoming requests are received by the dispatcher, and they are redirected to one of the member servers for actual processing. *Id.* at Section 3.1.

189. Garland is agnostic as to the type of processing done by the member servers, which can include dynamic Web page generation. Garland further discloses the use of dynamic load balancing to dispatch the web page requests from the dispatcher server to the member servers according to dynamic load information exchanged between the dispatcher server and the member servers. See A48 at Figure 5 and Section 3.4 titled "Load Balancing." Because Garland

also teaches the use of HTTP to transmit web page requests, there would exist, according to epicRealm, TCP releasing between the member servers and the dispatcher server.

190. Garland teaches that it would be obvious to combine its distributed Web server architecture and load balancing techniques with any system (e.g., a two-tier or three-tier web server architecture) that suffers from the drawbacks identified in the Introduction, namely, the problem of having a single server receiving and processing too many requests. Garland teaches that requests can be divided up and dispatched among two or more distributed and parallel servers such that server load can be spread out among many machines that have independent processing power and such that the front-end server (e.g., a web server or dispatcher server) and the several member servers (e.g., page servers) can concurrently process requests. See, generally, *id.* at Section 3.

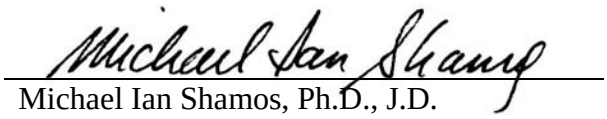
191. Garland also introduces a motivation to combine historic load balancing techniques with the management of web page requests in a web architecture such as that disclosed in Garland's Figure 5. Particularly, Section 8 titled "Bibliography" cites to a variety of load balancing prior art references that describe load balancing techniques in various system and server architectures that are not necessarily in the field of Internet technology. Garland, therefore, demonstrates a motivation to combine traditional system load balancing techniques with more modern server architectures that are used for the management of web page requests including dynamic web page requests. Accordingly, as taught by Garland, it would be obvious to one of ordinary skill in the art to combine any of the above prior art references discussed above with Garland. Such a combination would result in the above prior art references having the dispatching capabilities of Garland, namely the use of dynamic load balancing techniques to dispatch Web requests from a Web server to one or more page servers.

192. Notwithstanding the fact that all of the Asserted Claims are anticipated by the

above prior art references, the Asserted Claims are also obvious in view OWS 1.0, OWS 2.0, Dienst, or Popp in combination with Garland.

VII. CONCLUSION

I declare under penalty of perjury under the laws of the United States of America that the foregoing is true and correct and that I executed this Declaration [as originally filed] on the 28th day of July 2008, at Pittsburgh, Pennsylvania, and this Declaration [as corrected as to paragraph numbers only] on this 19th day of August 2008, at Texarkana, Texas.



Michael Ian Shamos, Ph.D., J.D.

61423565 v1

CERTIFICATE OF SERVICE

I hereby certify that on August 20, 2008, I caused the foregoing to be electronically filed with the Clerk of the Court using CM/ECF which will send electronic notification of such filing to the following:

Richard L. Horwitz
David Ellis Moore
POTTER ANDERSON & CORROON, LLP

Additionally, I hereby certify that true and correct copies of the foregoing were caused to be served on August 20, 2008 upon the following individuals in the manner indicated:

BY E-MAIL AND HAND DELIVERY

Richard L. Horwitz
David Ellis Moore
POTTER ANDERSON & CORROON, LLP
1313 N. Market St., Hercules Plaza, 6th Floor
P.O. Box 951
Wilmington, DE 19899-0951

rhorwitz@potteranderson.com
dmoore@potteranderson.com

BY E-MAIL

George S. Bosy
JENNER & BLOCK
330 N. Wabash Avenue
Chicago, IL 60611-7603

gbosy@jenner.com

/s/ James W. Parrett, Jr.
James W. Parrett, Jr. (#4292)

61423565 v1