IN THE UNITED STATES DISTRICT COURT
FOR THE DISTRICT OF DELAWARE

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

ORACLE'S BRIEF IN SUPPORT OF ITS MOTION FOR SUMMARY JUDGMENT OF INVALIDITY OF THE ASSERTED CLAIMS

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
mgraham@mnat.com
jparrett@mnat.com
(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

DATED: July 31, 2008
Redacted Filing Date: August 7, 2008

TABLE OF CONTENTS

Page

I. NATURE AND STAGE OF THE PROCEEDINGS 1

II. SUMMARY OF ARGUMENT 2

III. STATEMENT OF FACTS 2

A. EpicRealm's Claimed Inventions Were Well Known in the Prior Art Before the Filing and Alleged Invention Dates of the Patents-in-Suit 2

B. Beginning in 1994, Oracle Developed Its Own Prior Art Web Server Products Including Oracle WebServer 1.0 and 2.0 4

C. During Reexamination of the Patents-in-Suit, the Patent Office Has Found the Asserted Claims Invalid in View of Multiple Prior Art References 6

D. [redacted]

IV. LEGAL STANDARDS 8

A. Summary Judgment 8

B. Invalidity Under 35 U.S.C. § 102 – Anticipation 8

C. Invalidity Under 35 U.S.C. § 103 – Obviousness 8

V. ARGUMENT 9

A. The Asserted Claims Are Invalid as Anticipated by Oracle WebServer 1.0 9

1. Oracle WebServer 1.0 is Prior Art Under 35 U.S.C. §§ 102(a) and (g) 9

2. Oracle WebServer 1.0 Discloses Every Limitation of Every Asserted Claim 9

B. The Asserted Claims Are Invalid as Anticipated by Oracle WebServer 2.0 18

1. Oracle WebServer 2.0 is Prior Art Under 35 U.S.C. §§ 102(a) and (g) 18

2. Oracle WebServer 2.0 Discloses Every Limitation of Every Asserted Claim 18

C. The Asserted Claims Are Invalid as Anticipated by Popp 26

1. The Popp Reference is Prior Art Under 35 U.S.C. § 102(a) 26

2. Popp Discloses Every Limitation of Every Asserted Claim 26

D. The Asserted Claims Are Invalid as Anticipated by Dienst 33

1. The Dienst Reference is Prior Art Under 35 U.S.C. § 102(a) 33

2. Dienst Discloses Every Limitation of Every Asserted Claim 33

E. In the Alternative, the Asserted Claims Are Invalid Under § 103 as Obvious in View of Each Above Prior Art Reference in Combination with Garland 38

1. Garland is Prior Art under 35 U.S.C. § 102(b) 38

2. Garland's Disclosure of "Dispatching" Can Be Combined with any of the Anticipation References as an Alternative Bases for Invalidity 38

VI. CONCLUSION 40

TABLE OF AUTHORITIES

Page

Cases

Johnston v. IVAC Corp.
885 F.2d 1574 (Fed. Cir. 1989) 8

KSR Int'l Co. v. Teleflex, Inc.
127 S. Ct. 1727 (2007) 9

Medichem, S.A. v. Rolabo, S.L.
353 F.3d 928 (Fed. Cir. 2003) 8

Scanner Techs. Corp. v. Icos Vision Sys. Corp. N.V.
528 F.3d 1365 (Fed. Cir. 2008) 8

Scripps Clinic & Res. Found. v. Genentech, Inc.
927 F.2d 1565 (Fed. Cir. 1991) 8

Telemac Cellular Corp. v. Topp Telecom, Inc.
247 F.3d 1316 (Fed. Cir. 2001) 8

Statutes

35 U.S.C. § 102(a) 13

35 U.S.C. §§ 102(a) and (g) 2, 9, 18

35 U.S.C. § 102(b) 38

35 U.S.C. § 102(e) 2

35 U.S.C. § 103 ..... 8, 38

35 U.S.C. § 103(a) 2

Other Authorities

Fed. R. Civ. P. 56(c) 8

Plaintiffs and Counterdefendants Oracle Corporation and Oracle U.S.A. Inc. (jointly "Oracle") submit this brief in support of Oracle's Motion for Summary Judgment of Invalidity.

I. NATURE AND STAGE OF THE PROCEEDINGS

Oracle incorporates by reference its summary of the Nature and Stage of Proceedings set forth in its Opening Claim Construction Brief, filed herewith.[1]

Defendant epicRealm Licensing LP ("epicRealm") asserts in this case that Oracle infringes claims 1-5 and 7-11 of U.S. Patent No. 5,894,554 and claims 2 and 16 of U.S. Patent No. 6,415,335 (the "asserted claims" of the "patents-in-suit"). Prior to and during discovery, Oracle identified numerous prior art references that anticipate or render obvious all of the asserted claims of the patents-in-suit in view of epicRealm's overly broad reading of the patents' claims. Almost all these references were not considered by the Patent Office during its initial examination of the applications that issued as the patents-in-suit. During its recent reexamination of the patents-in-suit, the Patent Office rejected all the claims of the '554 patent as anticipated by at least six different prior art references. This motion demonstrates why the asserted patent claims are invalid in view of epicRealm's application of both parties' proposed claim constructions to Oracle's accused products. Particularly, the asserted claims are anticipated or obvious in view of two prior art references on which the Patent Office has relied to reject the asserted claims and two prior art products (Oracle WebServer 1.0 and 2.0) that Oracle itself developed and commercialized before the patents-in-suit were filed on April 23, 1996.

---

[1] In addition to the present motion, Oracle also files herewith: 1) Oracle's Opening Claim Construction Brief; 2) Oracle's Motion for Summary Judgment of Non-Infringement; 3) Oracle's Motion for Partial Summary Judgment of No Willful Infringement; 4) Oracle's Motion to Exclude Expert Report and Testimony of epicRealm's Damages Expert, Michael J. Wagner; and 5) Oracle's Motion to Exclude Damages Based on Foreign Sales. Exhibits in support of Oracle's motions are contained in the Appendix of Exhibits in Support of Oracle's Motions. Evidentiary support for each of the exhibits is contained in the accompanying declarations in support of Oracle's motions.

II. SUMMARY OF ARGUMENT

1. The asserted claims are invalid under 35 U.S.C. §§ 102(a) and (g) as being anticipated by the Oracle WebServer 1.0 and 2.0 products.

2. The asserted claims are invalid under 35 U.S.C. § 102(e) as being anticipated by U.S. Patent No. 6,249,241 issued to Popp et al. ("Popp").

3. The asserted claims are invalid under 35 U.S.C. § 102(a) as being anticipated by Lagoze et al., Dienst: *Implementation and Reference Manual*, Cornell University Technical Report TR95-1514 (May 5, 1995) ("Dienst").

4. In the alternative, the asserted claims are invalid under 35 U.S.C. § 103(a) as being obvious in view of any of the above anticipating references in combination with Garland et al., *Implementing Distributed Server Groups for the World Wide Web*, Carnegie Mellon University, Technical Report CMU-CS-95-114 (January 25, 1995) ("Garland").

III. STATEMENT OF FACTS

The Statement of Facts in Oracle's opening brief on Claim Construction, which Oracle incorporates by reference, contains background on the parties and general facts of the case.

A. EpicRealm's Claimed Inventions Were Well Known in the Prior Art Before the Filing and Alleged Invention Dates of the Patents-in-Suit

The '554 patent application was filed on April 23, 1996. *See* Oracle's "Appendix of Exhibits" (cited hereafter as "A_") at A1 ['554 patent]. The '335 patent application was later filed as a continuation to the '554 patent and, therefore, both patents share the same specification and filing date. A2 ['335 patent]. In general, the patents-in-suit disclose and claim methods and apparatus for managing dynamic Web page generation requests. A *dynamic* Web page, in contrast to a *static* Web page, is a Web page that a server typically generates anew every time the page is requested by a web client (*e.g.*, a Web browser) because the information contained in the

page changes frequently over time. Declaration of Dr. Michael Ian Shamos in Support of Oracle's Motion for Summary Judgment of Invalidity ("Shamos Decl.") at ¶¶21-23. One example of a dynamic Web page is a page that reports a set of stock quotes. Because stock prices can change minute to minute, Web pages that report such information are typically generated dynamically at the time they are requested using the most up-to-date data.

EpicRealm does not assert it invented Web servers or dynamic Web pages. Indeed, a Web server's generation of such pages was well known in the prior art, especially to Oracle, as early as 1994. Declaration of Katrina Montinola ("Montinola Decl.") at ¶3; Shamos Decl. at ¶39. EpicRealm instead asserts it invented a particular method and system architecture for managing dynamic Web page requests. That architecture is shown in Figure 4 of the patents-in-suit. The alleged inventions of epicRealm's patents-in-suit, however, were also well known in the prior art, especially in view of epicRealm's broad reading of the claims and application of the parties' proposed claim constructions.[2] Shamos Decl. at ¶¶41-42.

The independent claims of the patents-in-suit are generally directed to a method (or an apparatus for practicing a method) for handling dynamic Web page requests including (1) intercepting a request at a Web server, (2) dispatching that request to a particular page server, (3) releasing the Web server to process other requests, and (4) generating the requested dynamic Web page. *See, e.g.*, A1 at claim 1. As discussed in the patents' specification, the goal of the claimed inventions is to offload the processing of dynamic Web page requests from the Web server to software running on a separate computer (the page server), thus reducing the processing

---

[2] EpicRealm asserts its claimed inventions are entitled to an invention date earlier than the August 23, 1996 filing date because such inventions were allegedly conceived in August 1995 and diligently reduced to practice. epicRealm, however, has failed to present sufficient evidence during discovery showing that it is entitled to such an early date. Shamos Decl. at ¶42 and fn.2.

load on the Web server. *Id.* at col. 4:33-54. This concept, however, of spreading out server processing across multiple computers situated across separate tiers was well known in the art even before the Internet boom of 1995. Shamos Decl. at ¶¶25-33. Additionally, the claimed step of "dispatching" a request to a particular server (*e.g.*, a page server), selected from among several servers, was also well known. *Id.* at ¶¶31 and 38.

At the time of the asserted claims' alleged invention, the Internet community, including Oracle, had already devised identical solutions to the problems purported to be solved by the patents' applicants. *Id.* at ¶¶36-41. Under epicRealm's application of the parties' claim constructions, these early solutions anticipated each and every asserted claim.

B. Beginning in 1994, Oracle Developed Its Own Prior Art Web Server Products Including Oracle WebServer 1.0 and 2.0

Founded in 1977 by Silicon Valley software engineers Larry Ellison, Bob Miner, and Ed Oates, Oracle is the world's leading supplier of software for information management, and the world's second largest independent software company. Oracle specializes in developing and marketing enterprise software products, particularly database management systems, middleware including internet application servers, and business applications. In 1995, Oracle was no stranger to the Internet. Oracle was, to the contrary, a pioneer in integrating World Wide Web innovation with database technology to make information management more powerful than ever. Today, Oracle is the first software company to develop and deploy 100 percent internet-enabled enterprise software across its entire product line.

In 1994, Oracle software engineers began researching ways to use the Internet to enhance its database products. Montinola Decl. at ¶3. By early 1995, Oracle released its first web-based product called the Web-Oracle-Web ("WOW") Gateway. *Id.* The WOW Gateway made it possible for a web server to connect to an Oracle database, retrieve data and generate dynamic

Web pages containing the retrieved data. *Id.* Oracle made the WOW Gateway available to the public for free, and it was quickly put into use at various locations around the world. *Id.*

In early 1995, while the Oracle WOW Gateway was being developed, Oracle engineers conceived of a more sophisticated web-based product called Oracle WebServer 1.0 ("OWS 1.0"). Montinola Decl. at ¶4. OWS 1.0 together with the then well-known Oracle7 Server (a database server) comprised a system for providing access to database information over the Internet. *Id.* OWS 1.0 included a Web Listener for interfacing with a client (*e.g.*, a web browser) and a Web Agent for dispatching dynamic Web page requests to an Oracle7 Server. *Id.* Client requests are received at the Web Listener, which is a web server capable of handling requests for both static and dynamic web pages. *Id.* Requests for dynamic web pages are intercepted at the Web Listener and are routed to the Web Agent. *Id.* The Web Agent examines the request to determine which database server (or which application on the database server) should process the request. *Id.* The Web Agent then logs into the appropriate Oracle7 Server and dispatches the request to the appropriate application for the generation of the requested dynamic Web page. *Id.*

Oracle worked diligently to reduce OWS 1.0 to practice by July 1995, and the beta version of the product was made accessible to the public in September 1995. *Id.* at ¶¶5-7. The beta release represented a finished product, and its core architecture and operation were identical to that of the first commercial release of OWS 1.0 which occurred in October 1995. *Id.* at ¶¶6-9.

While Oracle was developing OWS 1.0 it was concurrently developing its next generation product, OWS 2.0. Montinola Decl. at ¶10. OWS 2.0 operated similarly to OWS 1.0 but also included a few enhancements. For example, OWS 2.0 included a Web Request Broker ("WRB") Dispatcher that dispatched dynamic Web page generation requests to any of a variety of WRB Services (*i.e.*, page servers) including, for example, a PL/SQL Agent, a Java Interpreter, and a LiveHTML Interpreter. *Id.* These page servers used different software languages (*e.g.*,

PL/SQL, Java, and LiveHTML) to generate dynamic web pages. Shamos Decl. at ¶82. Additionally, the WRB Dispatcher used dynamic load balancing to assist it in selecting efficient page servers for processing the requests. *Id.* at ¶88; Montinola Decl. at ¶10.

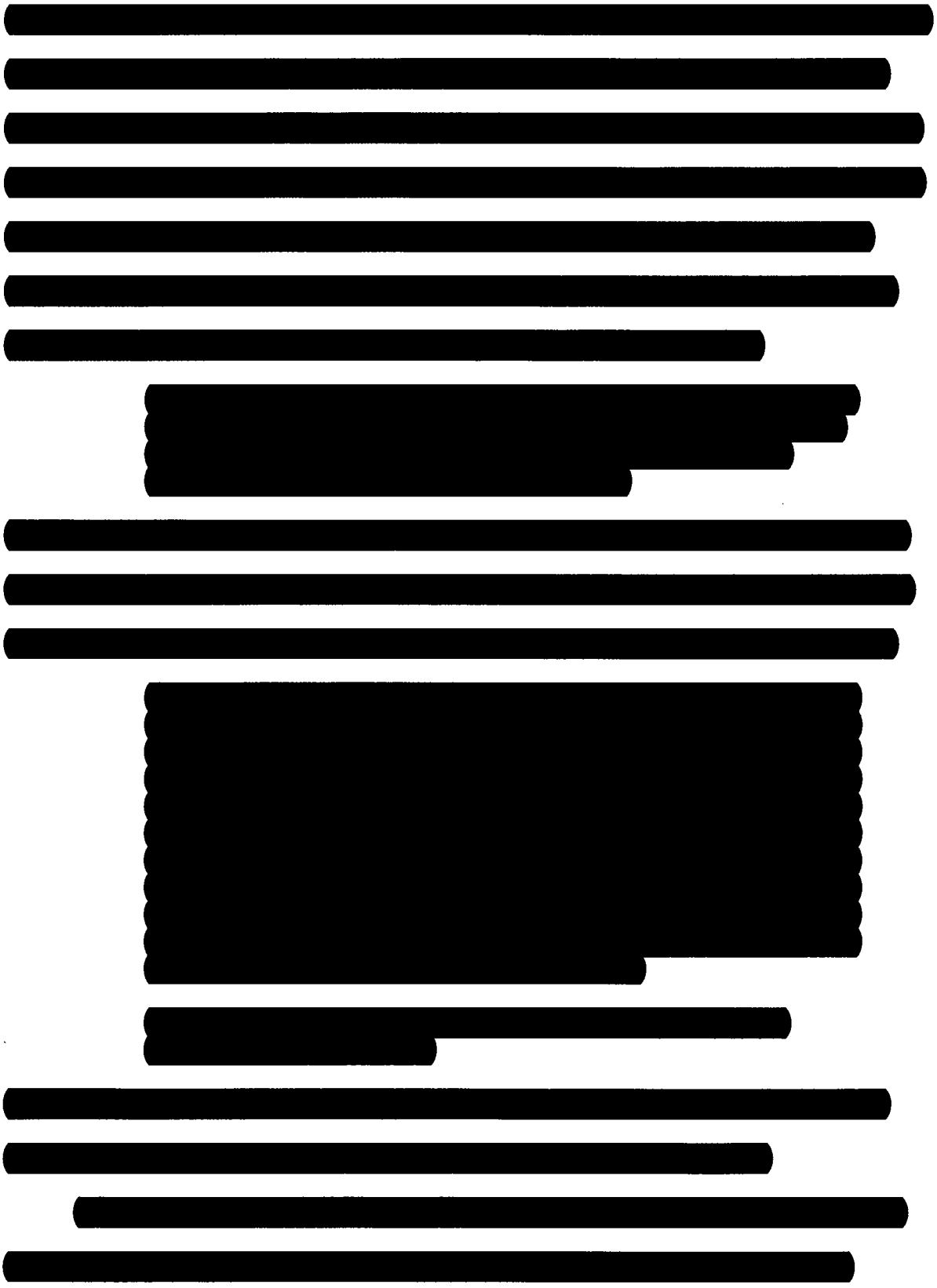
The core components of OWS 2.0 -- including the Web Listener (the Web server), the Web Request Broker (the dispatcher), and the WRB Services (the page servers) -- were conceived by Oracle no later than July 1995, and Oracle worked diligently to reduce OWS 2.0 to practice by October 1995. *Id.* at ¶¶11-12. Later, in February 1996, Oracle disclosed OWS 2.0 to the public at an E-mail World & Internet Expo in San Jose, California. *Id.* at ¶¶13-14.

Under epicRealm's application of the parties' proposed claim constructions, Oracle's WebServer 1.0 and 2.0 products anticipate each and every asserted claim of the patents-in-suit.

C. During Reexamination of the Patents-in-Suit, the Patent Office Has Found the Asserted Claims Invalid in View of Multiple Prior Art References

Between November 2006 and April 2007, three separate parties, one being Oracle, filed requests for the Patent Office to reexamine the patents-in-suit. The Patent Office granted all six requests (three for each patent) finding that they raised substantial new questions of patentability for every claimed invention. Declaration of Theodore T. Herhold ("Herhold Decl.") at ¶7. The reexamination requests were eventually consolidated into a single request for each patent-in-suit. *Id.* On June 30, 2008, the Patent Office issued an Office Action on its reexamination of the '554 patent rejecting *all eleven claims* as being anticipated by at least six prior art references. A10 at pp. A10-32 to 75. Two of those prior art references, including "Popp" and "Dienst," are discussed in detail below. *See* Summary of Argument, *supra*, for full cite and A45 and A46, respectively. The Patent Office has not yet acted on the '335 patent reexamination request.

D. [redacted]



IV. LEGAL STANDARDS

A. Summary Judgment

A grant of summary judgment pursuant to Rule 56(c) is appropriate "if the pleadings, depositions, answers to interrogatories, and admissions on file, together with affidavits, if any, show that there is no genuine issue as to any material fact and that the moving party is entitled to judgment as a matter of law." Fed. R. Civ. P. 56(c). This standard is applicable to patent cases. *Johnston v. IVAC Corp.*, 885 F.2d 1574, 1576-77 (Fed. Cir. 1989). Oracle bears the burden of proving invalidity by clear and convincing evidence. *Scanner Techs. Corp. v. Icos Vision Sys. Corp. N.V.*, 528 F.3d 1365, 1379 (Fed. Cir. 2008).

B. Invalidity Under 35 U.S.C. § 102 – Anticipation

In order to anticipate, a single prior art reference must disclose each and every limitation of the claimed invention, either expressly or inherently. *Telemac Cellular Corp. v. Topp Telecom, Inc.*, 247 F.3d 1316, 1327 (Fed. Cir. 2001). Although anticipation is a question of fact, it may be decided on summary judgment if there is no genuine dispute of material fact on the record. *Id.* There must be no difference between the claimed invention and the reference disclosure as viewed by a person of ordinary skill in the field of the invention. *See Scripps Clinic & Res. Found. v. Genentech, Inc.*, 927 F.2d 1565, 1576 (Fed. Cir. 1991). Determining whether the claims of a patent are anticipated is a two-step process. The claims are first interpreted to determine their scope. Next, the properly-construed claims are compared to the anticipatory reference. *Medichem, S.A. v. Rolabo, S.L.*, 353 F.3d 928, 933 (Fed. Cir. 2003).

C. Invalidity Under 35 U.S.C. § 103 – Obviousness

Patent claims are invalid as obvious when "the 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 by a person having ordinary skill in the art . . .

." *KSR Int'l Co. v. Teleflex, Inc.*, 127 S. Ct. 1727, 1734 (2007). A patent "composed of several elements is not proved obvious merely by demonstrating that each of its elements was, independently, known in the prior art." *Id.*, 127 S. Ct. at 1740-41. When a patent, however, "simply arranges old elements with each performing the same function it had been known to perform and yields no more than one would expect from such an arrangement, the combination is obvious." *Id.* at 1739-40 (internal quotations omitted). "In determining whether the subject matter of a patent claim is obvious, neither the particular motivation nor the avowed purpose of the patentee controls. What matters is the objective reach of the claim. If the claim extends to what is obvious, it is invalid under § 103." *Id.* at 1739, 1741-42.

V. ARGUMENT

A. The Asserted Claims Are Invalid as Anticipated by Oracle WebServer 1.0

1. Oracle WebServer 1.0 is Prior Art Under 35 U.S.C. §§ 102(a) and (g)

Oracle made OWS 1.0 accessible to the public no later than September 25, 1995. Montinola Decl. at ¶¶6-7. Because epicRealm cannot prove an invention date earlier than September 25, 1995, OWS 1.0 is prior art to the patents-in-suit under 35 U.S.C. § 102(a). Shamos Decl. at ¶42 and fn.2. Additionally, Oracle conceived of and diligently reduced to practice OWS 1.0 prior to epicRealm's earliest asserted invention date of August 1995, thus making OWS 1.0 also prior art under 35 U.S.C. § 102(g). Montinola Decl. at ¶¶4-6.

2. Oracle WebServer 1.0 Discloses Every Limitation of Every Asserted Claim

As demonstrated below, the Oracle WebServer 1.0 system (including the Oracle7 server) ("OWS 1.0") anticipates each and every asserted claim of the patents-in-suit.

> **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:**

OWS 1.0 meets claim 1's preamble because it managed dynamic Web page generation requests to a Web server (the "HTTP server" or "Web Listener" of OWS 1.0): "[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." A40 at ORCL000021; Shamos Decl. at ¶44.

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,

As shown in Figure 1 below, OWS 1.0 discloses a Web client (the web browser) sending a dynamic Web page request[3] to a Web server (the Oracle Web Listener).[4] *Id.* OWS 1.0 further discloses routing the request from the Web server to a page server (an Oracle7 Server) via a dispatcher (the Oracle Web Agent and SQL*Net). Shamos Decl. at ¶45.

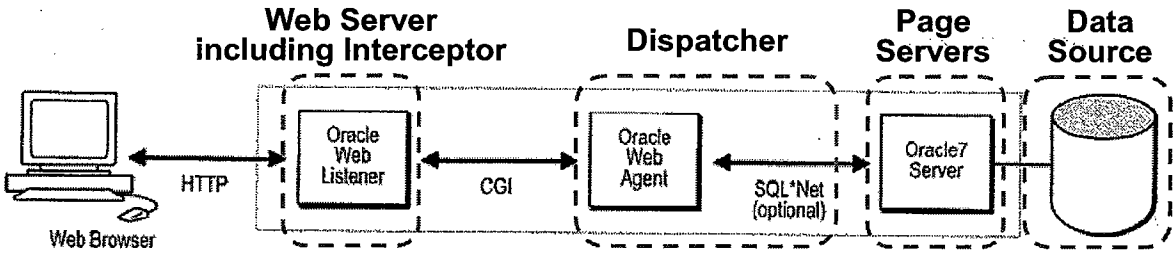


Figure 1 - from the OWS 1.0 User's Guide - A40 at ORCL000022

As discussed in the OWS 1.0 User's Guide, the Oracle7 Server is a "page server" of the patents-in-suit under both parties' proposed constructions for that term.[5] For example, the Oracle7 Server is a "page server" under epicRealm's proposed constructions because it includes

---

[3] This OWS 1.0 "request" meets both parties' proposed constructions for that term, which are listed in CHART 1 attached hereto. Shamos Decl. at ¶45 and fn.3.

[4] The OWS 1.0 Web Listener meets both parties' proposed constructions for "Web server," which are listed in CHART 1 attached hereto. Shamos Decl. at ¶45 and fn.4.

[5] The parties' proposed constructions for "page server" are listed in CHART 1 attached hereto.

software for generating dynamic HTML pages: "[t]he Oracle7 Server provides the storage for all dynamic data in relational tables, and all the program logic used to create dynamic HTML pages." A40 at ORCL000021; Shamos Decl. at ¶45. Similarly, the Oracle7 Server is a "page server" under Oracle's proposed construction for that term because as a database server it can run on a different machine (*i.e.*, different processor) from that of the Web Server. *Id.*

OWS 1.0 also discloses "said page server . . . ***releasing*** said Web Server to process other requests" under epicRealm's interpretation of both parties' proposed claim constructions.[6] Under epicRealm's implicit releasing theory,[7] for example, the mere fact that the Web Listener hands off requests to an Oracle7 server and continues processing new requests is sufficient to meet epicRealm's proposed "releasing" construction. Shamos Decl. at ¶46. Additionally, because OWS 1.0 passes Web page requests to an Oracle7 page server via HTTP and the TCP protocol, OWS 1.0 also discloses the page server releasing the Web server under epicRealm's TCP releasing theory.[8] *Id.*

> **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;**

---

[6] The parties' proposed constructions for "said page server . . . releasing said Web Server" are listed in CHART 1 attached hereto.

[7] Under its proposed "releasing" construction (*i.e.*, "freeing"), epicRealm argues that the mere fact that the Web Server hands off requests to the page server is sufficient to meet the claims' step of "releasing" because the hand off relieves (or frees) the Web server of the obligation of processing the request. Shamos Decl. at ¶46 and fn.7. Oracle strongly disputes that this implicit releasing meets the literal language of the claims. If the Court accepts epicRealm's implicit releasing theory, however, OWS 1.0 also discloses such releasing. *Id.*

[8] Under this theory, epicRealm asserts that under both parties' proposed constructions for "releasing," an accused page server "releases" the Web server to process other requests via a TCP acknowledgement ("TCP ACK") message sent from the page server to the Web server. Shamos Decl. at ¶46 and fn.8. Although Oracle strongly disputes that TCP ACK messages can "release" a Web server to process other requests, epicRealm has chosen to pursue this infringement theory. If the Court accepts epicRealm's TCP releasing theory, however, OWS 1.0 also discloses such releasing under both parties' proposed constructions. *Id.*

OWS 1.0 further discloses these steps. First, as discussed in the OWS 1.0 User's Guide, OWS 1.0 discloses intercepting[9] a request at the Web server (the Oracle Web Listener):

> 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 . . . .

A40 at ORCL000022; Shamos Decl. at ¶47. This act of distinguishing between static and dynamic page requests and diverting the dynamic page requests away from the Web server to another program (*e.g.*, a page server) meets both parties' proposed constructions for "intercepting." *Id.*

OWS 1.0 further discloses routing the request from the Web server to a dispatcher (an OWS Web Agent). Shamos Decl. at ¶48. The OWS Web Agent is a "dispatcher" and performs "dispatching" under both parties' proposed constructions for those terms.[10] For example, the Oracle Web Agent is a separate program that the Web Listener invokes when a request for a dynamic document is received. The Oracle Web Agent examines the request's URL and dynamic information maintained in a configuration file to make an informed selection of which Oracle7 Server (page server) to which to connect:

> 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. This information is stored in the owa.cfg file . . . as part of a Web Agent service. The Web Agent determines which service to use by parsing the `SCRIPT_NAME` environment variable, which is set up by the

---

[9] The parties' proposed constructions for "intercepting said request . . ." are listed in CHART 1 attached hereto.

[10] The parties' proposed constructions for "dispatcher" and "dispatching said request . . ." are listed in CHART 1 attached hereto.

> Web Listener . . . . The Web Lister derives the value of this environment variable from the path section just before the `/owa` in the URL.

A40 at ORCL000115; Shamos Decl. at ¶48. Under epicRealm's proposed construction for "dispatching," the Oracle Web Agent uses dynamic information stored in a configuration file to select the Oracle7 Server (a page server) that can more efficiently process the request. *Id.* 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. *Id.*

OWS 1.0 performs further dispatching of requests via SQL*Net under epicRealm's application of the parties' proposed claim constructions. Shamos Decl. at ¶¶49-52. SQL*Net[11] allows the Web Agent to dispatch requests to different page servers running on remote database servers. *Id.* at ¶¶50-51. After the Oracle Web Agent determines which Oracle7 server to which to connect, it sends a connection request to a Network Listener associated with that server. *Id.* The Network Listener makes an informed selection of which particular Oracle7 server process (or set of processes) should process the request. *Id.* The Listener can dispatch the Web page request to one of three types of server processes: (a) a dedicated server; (b) a multi-threaded server; or (c) a prespawned dedicated server. *Id.*; *see also* A49 at ORCL01806664. If the connection request directs the Listener to select a multi-threaded server, the page request is routed to one of several "Shared Dispatchers" for further processing. *Id.* The Listener also maintains dynamic information about the load on the various Shared Dispatchers and selects the Dispatcher that can more efficiently process the request. *Id.* Under epicRealm's application of both parties' proposed constructions, this informed selection based on dynamic information

---

[11] SQL*Net v. 2.2 is also prior art to the patents-in-suit under 35 U.S.C. § 102(a) because it was on sale to the public no later than early August 1995. *See* Herhold Decl. at ¶8; A54.

constitutes "dispatching." *Id.*; *see also* A59 at A59-3 (showing that epicRealm has admitted in discovery that a Listeners' routing of requests to a selected "Dispatcher" is "dispatching").

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

The Oracle7 server (the page server) processes the dynamic page generation request and returns the generated page to the Web Agent. Shamos Decl. at ¶53. This process occurs simultaneously with the Web Listener (the web server) processing other requests because, as discussed above, the Web Listener is "released," under epicRealm's application of the claims, to process other requests while the Oracle7 Server is processing the handed off request. *Id.*

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

The Oracle7 server (the page server) processes the dynamic page generation request and returns the generated Web page[12] to the Web Agent. Shamos Decl. at ¶54. The generated web page includes data dynamically retrieved from one or more data sources by means of PL/SQL procedures, which access an SQL database (*i.e.*, an Oracle7 database). *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.

OWS 1.0 further discloses the step of identifying one or more data sources.[13] For example, "[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; Shamos Decl. at ¶55. The data sources to be used are identified by examining the

---

[12] This Web page meets both parties' proposed constructions for "Web page," which are listed in CHART 1 attached hereto. Shamos Decl., ¶53 and fn.11.

[13] The parties' proposed constructions for "data sources" are listed in CHART 1 attached hereto.

PATH_INFO environment variable. *Id.*

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.

OWS 1.0 discloses dynamically retrieving said data from one or more data sources. As discussed above, the Oracle7 server enables the creation of dynamic Web pages from data stored in an Oracle database. Shamos Decl. at ¶56. "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." A40 at ORCL000021. The Oracle7 Server (page server) invokes PL/SQL procedures that "extract data from the Oracle7 database and generates an HTML document in a PL/SQL table." *Id.* at ORCL000116; Shamos Decl. at ¶56.

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.

OWS 1.0 can use connection caching[14] when the Oracle Web Agent connects to an Oracle7 Server (page server) with SQL*Net. Shamos Decl. at ¶57. For example, 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. *Id.*

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.

---

[14] The proposed constructions for "connection cache" are listed in CHART 1 attached hereto.

OWS 1.0 discloses logging into[15] said one or more data sources: "[i]n 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; Shamos Decl. at ¶58. The username and password are used to "log into" the data source according to Oracle's proposed construction for that term. *Id.*

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.

OWS 1.0 discloses page servers including custom HTML extension templates for configuring a Web page. OWS 1.0 provides a Developer's Toolkit incorporating hypertext procedures (HTP): "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. Shamos Decl. at ¶59.

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.

OWS 1.0 discloses inserting the dynamically retrieved data from 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. A40 at ORCL000151 and ORCL000124-126; Shamos Decl. at ¶60.

---

[15] The parties' proposed constructions for "logging into" are listed in CHART 1 attached hereto.

Claims 9 and 10 of the '554 patent - A networked system for managing a dynamic Web page generation request, said system comprising: . . .

Claims 9 and 10 of the '554 patent recite substantially the same limitations of claim 1 except that the claimed steps are performed on a network system. As shown in Figure 1 above, OWS 1.0 operated on a network of computers including a client computer (running the Web browser), the Web Server computer (running the Web Listener) and the page server computer (running the Oracle7 Server). *Id.* at ¶¶61-69. Claim 9 also requires a "page server processing means," a "first computer system including means for generating said request," and "a second computer system including means for receiving said request." OWS 1.0 requires these three structures in order to operate: (1) the page server processing means is the processor running the Oracle7 Server; (2) the first computer system is the system running the Web Browser, which generates the HTTP requests; and (3) the second computer system is the system running the Web server, which receives the HTTP requests. *Id.*

Additionally, as discussed above with respect to claims 1 and 9, OWS 1.0 discloses claim 10's limitations including an "interceptor" and a "dispatcher" running on the OWS 1.0 network. The interceptor is the OWS 1.0 Web server software that performs the act of intercepting, and the dispatcher is the software that runs the Oracle Web Agent and SQL*Net. *Id.* at ¶¶70-72. Accordingly, OWS 1.0 discloses all the limitations of claims 9 and 10.

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: . . .

Claim 11 of the '554 patent discloses substantially the same limitations of claim 1 except that the claimed steps are performed when stored instructions on a computer readable medium are executed by a computer system. Because OWS 1.0 is a programmed system whose instructions are on a machine readable medium (*e.g.*, a hard disk), OWS 1.0 discloses all the limitations of this claim. Shamos Decl. at ¶¶73-74. Additionally, Oracle's OWS 1.0 and Oracle7

Server products were sold to customers on a machine readable medium such as a CD ROM. Montinola Decl. at ¶6.

Claims 2 and 16 of the '335 patent (including limitations of Claim 1)

As discussed above with respect to claim 1 of the '554 patent, OWS 1.0 discloses all the limitations of these claims, which are substantially identical to that of claim 1. Shamos Decl. at ¶¶75-78. The only significant difference between the limitations of '554 patent claim 1 and that of these claims is that in claim 16 the term "HTTP-compliant device" of this claim is used in place of "Web server" of claim 1 of the '554 patent. As discussed above, the OWS 1.0 is an HTTP-compliant device because it can receive and process Web page requests sent from clients sent over the HTTP protocol. *Id.* at ¶77.

B. The Asserted Claims Are Invalid as Anticipated by Oracle WebServer 2.0

1. Oracle WebServer 2.0 is Prior Art Under 35 U.S.C. §§ 102(a) and (g)

Oracle disclosed OWS 2.0 to the public no later than February 21, 1996. Montinola Decl. at ¶13. Because epicRealm cannot prove an invention date earlier than February 21, 1996, OWS 2.0 is prior art to the patents-in-suit under 35 U.S.C. § 102(a). Shamos Decl. at ¶42 and fn.2. Additionally, Oracle conceived of OWS 2.0 no later than July 1995 and diligently reduced it to practice by October 1995. Montinola Decl. at ¶¶11-12. Because Oracle's OWS 2.0 conception date predates epicRealm's earliest asserted invention date of August 1995, OWS 2.0 is also prior art under 35 U.S.C. § 102(g).

2. Oracle WebServer 2.0 Discloses Every Limitation of Every Asserted Claim

As demonstrated below, the Oracle WebServer 2.0 system (including the Oracle7 server) ("OWS 2.0") anticipates all of the asserted claims of the patents-in-suit.

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:

OWS 2.0 managed dynamic Web page generation requests to a Web server (the "HTTP server" or "Web Listener" of OWS 2.0): "[u]sing the Oracle Web Listener, your web site can respond to client requests by generating HTML documents dynamically." A42 [OWS 2.0 User's Guide] at ORCL01816648; Shamos Decl. at ¶80.

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,

As shown in Figure 2 below, OWS 2.0 discloses a Web client (the web browser) sending a dynamic Web page request to a Web server (the HTTP Server). Shamos Decl. at ¶81. OWS 2.0 further discloses routing the request from the Web server to a page server (a Web Request Broker Executable Engine ("WRBX") together with a WRB Service) via a dispatcher (the WRB Dispatcher). *Id.*

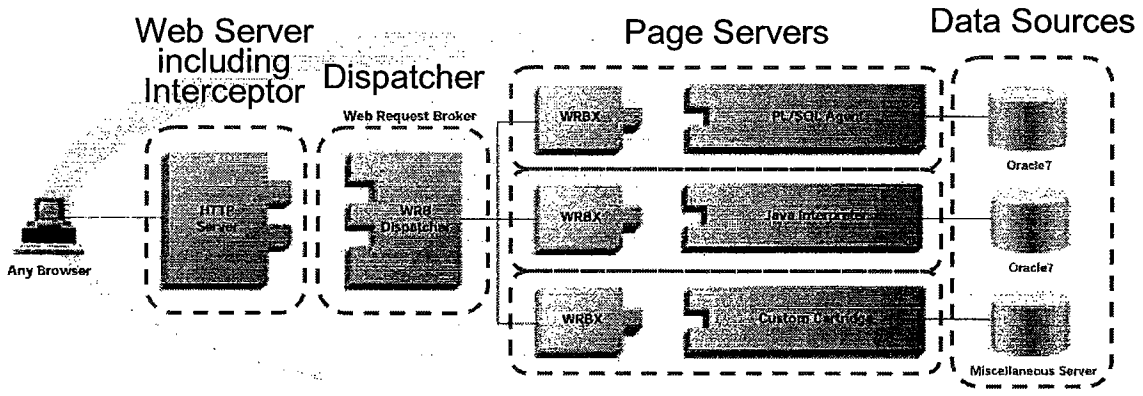


Figure 2 – A43 at ORCL000747

Like OWS 1.0, OWS 2.0 includes an HTTP Server (a Web Server) running an Oracle Web Listener for receiving Web page requests, processing requests, and returning Web pages in response to those requests: "[t]he Web Listener is the component that receives a URL from a Web browser and sends back the appropriate content" and "[u]sing the Oracle Web Listener, your web site can respond to client requests by generating HTML document dynamically." A42

at ORCL01816643 and 648; Shamos Decl. at ¶81.

The Web Listener is also responsible for routing dynamic Web page requests to a page server by means of a Web Request Broker ("WRB") Dispatcher. After the Web Listener hands off requests to the WRB, the WRB Dispatcher dispatches the requests to WRBX processes (page servers) that receive the requests and invoke server extensions called WRB Services. Shamos Decl. at ¶82. 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.* A WRBX process is an instance of a WRB Service and is a "page server" under epicRealm's proposed constructions for that term because it includes software for generating "dynamic HTML pages." *Id.* Similarly, a WRBX process is a "page server" under Oracle's proposed construction for that term because, for example, it can invoke stored PL/SQL procedures on a database running on a different machine (*i.e.*, different processor) from that of the Web Server, just as in the case of OWS 1.0, for the purpose of generating dynamic Web pages. *Id.*

Additionally, OWS 2.0 discloses "releasing said Web Server to process other requests" under epicRealm's application of both parties' proposed constructions. First, under epicRealm's implicit releasing infringement theory, the Web server is "freed" to process requests because the page sever receives the request and relieves the Web Server of the obligation to process it. *See* fn.7 *supra*; Shamos Decl. at ¶85. Additionally, under epicRealm's explicit releasing theory, OWS 2.0 discloses several means by which the WRBX processes (the page servers) perform acts that, according to epicRealm, release the Web server to process other requests. *See* fn.8 *supra*; Shamos Decl. at ¶¶83-84. For example, according to epicRealm's explicit releasing theory, the OWS 2.0 page servers release the Web server via Socket-Bind-Listen actions and by sending INIT and REQ COMPLETE Responses that, according to epicRealm, free up certain resources that would either allow processing of other requests or facilitate more efficient processing of

already received requests. *Id.*

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;

OWS 2.0 further discloses the step of intercepting said request at said Web server: "[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." A44 at ORCL000764; Shamos Decl. at ¶86. For example, the Web Request Broker may be configured to receive and dispatch all URL requests beginning with /java to the integrated Java Interpreter. Shamos Decl. at ¶87. This act of distinguishing between static page requests and dynamic page requests and diverting the dynamic page requests away from the Web server to another program (*i.e.*, a page server) meets both parties' proposed constructions for "intercepting." *Id.*

As shown above in Figure 2, OWS 2.0 further discloses routing a request from the Web server to a dispatcher (the WRB Dispatcher). The WRB Dispatcher is a "dispatcher" and performs "dispatching" under both parties' proposed constructions for those terms. *See* proposed constructions at CHART 1 attached hereto; Shamos Decl. at ¶¶88-89. Particularly, "[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." A44 at ORCL000766. 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 maintains continuously, so that HTTP requests requiring the execution of programs are not slowed down by the performance cost of spawning a new process." A42 at ORCL01816808. 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." A44 at ORCL0000766. 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 (*e.g.*, 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. Shamos Decl. at ¶89. 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. *Id.*

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

Concurrent processing between the Web server and the page server is taught expressly in the OWS 2.0 User's Guide: "[i]f 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." A42 at ORCL01816643. Because the Listener proceeds to listen for and process other requests and does not wait for the WRB Dispatcher or WRBX process to finish, it is capable of concurrently processing requests. Shamos Decl. at ¶90.

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

OWS 2.0 discloses this final limitation as well. As discussed above, the WRB Service instances process the dynamic page generation requests and return the generated Web page[16] to the Web server: "the Oracle Web Listener takes the contents of standard output and returns it to the Web browser that requested the dynamic HTML document." A42 at ORCL01816707;

---

[16] This Web page meets both parties' proposed constructions for "Web page," which are listed in CHART 1 attached hereto. Shamos Decl. at ¶91.

Shamos Decl. at ¶¶91-92. Additionally, a Web page generated by OWS 2.0 includes data dynamically retrieved from one or more data sources: "[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 . . . and the database is used for Web pages that are generated at runtime using 'live' data." A42 at ORCL01816643; Shamos Decl. at ¶93.

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.

OWS 2.0 discloses the step of identifying one or more data sources from which to retrieve said data using Database Connection Descriptors ("DCDs"). OWS 2.0 examines DCDs and their associated environment variables to identify and make connections to appropriate data sources. *See* "data source" constructions at CHART 1 attached hereto; Shamos Decl. at ¶94.

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.

OWS 2.0 discloses dynamically retrieving said data from one or more data sources. For example, "[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 . . . and the database is used for Web pages that are generated at runtime using 'live' data." A42 at ORCL01816643; Shamos Decl. at ¶95. Accordingly, OWS 2.0 dynamically retrieves data from a data source when it accesses the database at runtime to retrieve "live" data.

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.

OWS 2.0 discloses connection caching. For example, "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 ORCL000771-772; Shamos Decl., ¶96.

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.

OWS 2.0 discloses logging into said one or more data sources. For example, "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." A44 at ORCL000772. This form of logging into the database can involve presenting credentials for permission to access the database. Shamos Decl. at ¶97.

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.

OWS 2.0 discloses page servers including custom HTML extension templates for configuring a Web page. For example, OWS 2.0 discloses such extension templates with the use of the LiveHTML Interpreter (a WRB Service). *Id.* at ¶98. "The LiveHTML Interpreter enables you to include dynamic content in otherwise static Web pages." A42 at ORCL01816656.

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.

OWS 2.0 also discloses inserting the dynamically retrieved data from one or more data sources into said custom HTML extension templates. Again, OWS 2.0 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." A42 at ORCL01816698; Shamos Decl. at ¶99.

Claims 9 and 10 of the '554 patent - A networked system for managing a dynamic Web page generation request, said system comprising: . . .

Claims 9 and 10 of the '554 patent recite substantially the same limitations of claim 1 except that the claimed steps are performed on a network system. As shown in the Figure 2 above, OWS 2.0 operated on a network of computers including, at a minimum, a client computer and the computers running everything else including the Web server, the page servers and the data sources. Shamos Decl. at ¶¶100-109. Claim 9 also requires a "page server processing means," a "first computer system including means for generating said request," and "a second computer system including means for receiving said request." OWS 2.0 requires these three structures in order to operate: (1) the page server processing means is the processor running the Oracle7 Server; (2) the first computer system is the system running the Web Browser, which generates the HTTP requests; and (3) the second computer system is the system running the Web server, which receives the HTTP requests. *Id.*

Additionally, as discussed above with respect to claims 1 and 9, OWS 2.0 discloses claim 10's limitations including an "interceptor" and a "dispatcher" running on the OWS 2.0 network. The interceptor is the OWS 2.0 Web server software that performs the act of intercepting, and the dispatcher is the software that runs the WRB Dispatcher. *Id.* at ¶¶110-112. Accordingly, OWS 2.0 discloses all the limitations of claims 9 and 10.

> **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: . . .**

Claim 11 of the '554 patent discloses substantially the same limitations of claim 1 except that the claimed steps are performed when stored instructions on a computer readable medium are executed by a computer system. Because OWS 2.0 is a programmed system whose instructions are on or are downloaded from a machine readable medium (e.g., a hard disk or CD), OWS 2.0 discloses all the limitations of this claim. *Id.* at ¶¶113-114.

Claims 2 and 16 of the '335 patent (including limitations of Claim 1)

As discussed above with respect to claim 1 of the '554 patent, OWS 2.0 discloses all the limitations of these claims, which are substantially identical to that of claim 1. The only significant difference between the limitations of '554 patent claim 1 and that of these claims is that claim 16 requires a "HTTP-compliant device" instead of a "Web server." As discussed above, the OWS 2.0 is an HTTP-compliant device because it can receive and process Web page requests sent from clients sent over the HTTP protocol. *Id.* at ¶¶115-119.

C. The Asserted Claims Are Invalid as Anticipated by Popp

1. The Popp Reference is Prior Art Under 35 U.S.C. § 102(a)

U.S. Patent No. 6,249,291 ("Popp") discloses a method and apparatus for managing Internet transactions. A45. The Popp patent has a filing date of September 22, 1995. Because epicRealm cannot prove an invention date earlier than September 22, 1995, Popp is prior art to the patents-in-suit under 35 U.S.C. § 102(e). Shamos Decl. at ¶42 and fn.2.

2. Popp Discloses Every Limitation of Every Asserted Claim

As demonstrated below, Popp, including the system described therein, anticipates all the asserted claims of the patents-in-suit. Indeed, epicRealm's own expert, Dr. David Finkel, has admitted that Popp discloses all the limitations of '554 patent, claim 1 except "dispatching said request to said page server" under epicRealm's proposed claim construction for that term. A25 [Finkel Depo.] at pp. 89:22-90:3 and 100:13-24. As shown below, however, Popp does in fact disclose all of claim 1's limitations under both parties' proposed claim constructions.

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:**

Popp discloses a computer-implemented method for producing hypertext pages (dynamic

web pages) in response to queries (requests). Shamos Decl. at ¶120. 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 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,

As shown in Figure 3 below, for example, Popp discloses a Web client (Client Browser 202) sending a dynamic Web page request to a Web server (HTTP Server 206). A45 at MAC003930; Shamos Decl. at ¶121. Popp further discloses routing the request from the Web server to a page server (Application 214) via a dispatcher (a CGI interface program). *Id.*

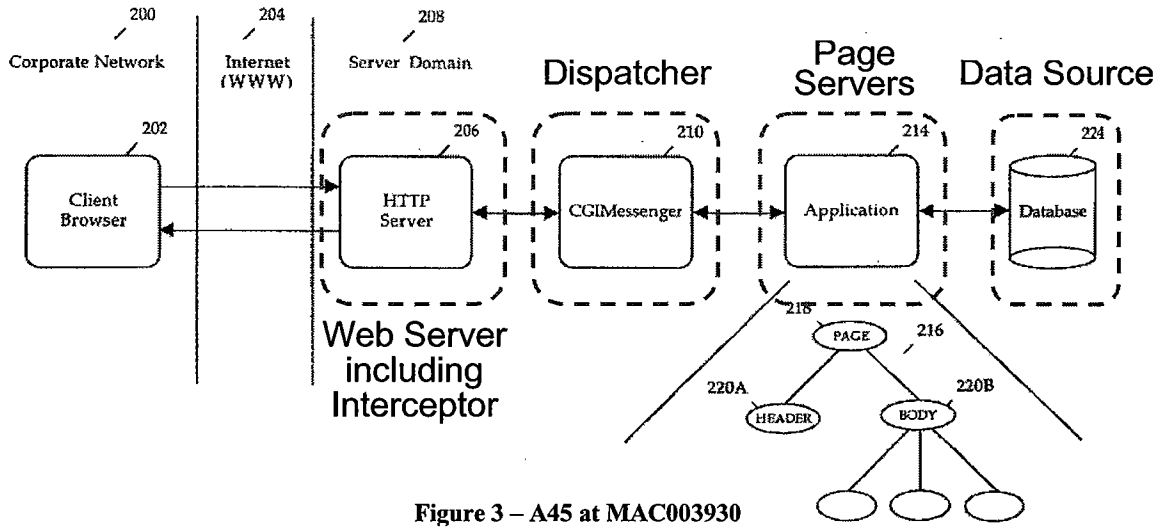


Figure 3 – A45 at MAC003930

Popp's HTTP Server is a "Web server" under both parties' proposed constructions because it is a server running HTTP-compliant software for receiving Web page requests and returning Web pages in response to those requests. A45 at col. 8:20-21; Shamos Decl. at ¶121.

Additionally, a Popp application 214 is a "page server" under epicRealm's proposed construction for that term because it includes software for generating dynamic Web pages.

Shamos Decl. at ¶122. 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.

A45 at col. 7:52-58. Similarly, 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 the Web server. A45 at col. 7:28-30; Shamos Decl. at ¶122.

Additionally, Popp discloses "releasing said Web Server to process other requests" under epicRealm's application of both parties' proposed constructions for that claim term. First, under epicRealm's implicit releasing infringement theory, the Web server is "freed" to process requests by virtue of the fact that the page sever receives the request and relieves the Web Server of the obligation to process it. Shamos Decl. at ¶124. Additionally, under epicRealm's explicit releasing infringement theory, Popp discloses several means by which the applications 214 (the page servers) perform acts that, according to epicRealm, release the Web server to process other requests. *Id.* For example, the Popp page servers release the Web server by sending TCP ACK messages and via Socket-Bind-Listen actions that, according to epicRealm's "TCP releasing" theory, free up certain resources that would either allow processing of other requests or facilitate more efficient processing of already received requests. *Id.* at ¶123.

> **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;**

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:

> 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 . . . 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. Requests that are intended for an application 214 are intercepted at the HTTP server and are routed to a CGI interface program (*e.g.*, CGIMessenger 210), and other requests (e.g., static requests) can be handled normally by the HTTP server. Shamos Decl. at ¶125. This act of distinguishing between static page requests and dynamic page requests and diverting the dynamic page requests away from the Web server to a page server meets both parties' proposed constructions for "intercepting." *Id.*

Popp further discloses routing a request from the Web server (HTTP server 206) to a dispatcher. Popp's CGI interface program is a "dispatcher" and performs "dispatching" under both parties' proposed constructions for those terms. *Id.* at ¶126. For example, Popp discloses dispatching with respect to a CGI interface program called a Web Objects Server:

> 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 continues to step 506.
>
> At step 506, the request is forwarded to the applications.

A45 at col. 27:42-55 (emphasis added). Popp also discloses dispatching with respect to another

CGI interface program called a CGIMessenger. A45 at col. 7:16-35; Shamos Decl. at ¶126.

As described above, a CGI interface program (*e.g.*, a Web Objects server or a CGIMessenger) examines the incoming request and dispatches it based on dynamic information maintained about the applications (*e.g.*, information relating to which application should process the request and on which server the application is executing). *Id.* Accordingly, Popp's CGI interface program 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 214) should process requests based on dynamic information maintained about page servers. 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. *Id.* Indeed, epicRealm's own expert, Dr. Finkel, admits that Popp discloses "dispatching" under Oracle's proposed construction for that term. A25 at p. 100:13-24.

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

Popp discloses concurrent processing between the Web server and the page server because Popp teaches that the HTTP server and the application servers can run on different machines (*i.e.*, different processors). Shamos Decl. at ¶127.

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

As discussed above, Popp discloses dynamically generating a Web page in response to the request which includes data dynamically retrieved from one or more data sources. A45 at col. 8:44-46; Shamos Decl. at ¶128.

<u>Claims 2 and 3 of the '554 Patent</u> - "identifying one or more data sources" and "dynamically retrieving data" from those sources

Popp discloses the steps of these claims including identifying one or more data sources from which to retrieve said data and dynamically retrieving data from those sources. Shamos

Decl. at ¶¶129-130. For example, Popp discloses that a CGI program can identify a Web Object server based on the information contained in the request, which can include a data source from which to retrieve data. A45 at col. 26:3-4. Additionally, an application server can dynamically retrieve data from those data sources to 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 - maintaining a "connection cache"

Popp discloses maintaining a connection cache to said one or more data sources. Shamos Decl. at ¶131; A45 at col. 8:28-31 and col. 13:38-40.

Claim 5 of the '554 patent - "logging into said one or more data sources"

Popp discloses logging into said one or more data sources using a digital signature to verify that a client is authorized to access data from a data source. Shamos Decl. at ¶132; A45 at col. 27:12-17.

Claims 7 and 8 of the '554 patent - "custom HTML extension templates" and "inserting said dynamically retrieved data" into said templates

Popp discloses the use of custom HTML extension templates: "[t]he 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." Shamos Decl. at ¶133; A45 at col. 15:45-50. Popp also discloses inserting dynamically retrieved data from one or more data sources into the custom HTML extension templates. Shamos Decl. at ¶134; A45 at col. 18:57-59.

Claims 9 and 10 of the '554 patent - A networked system for managing a dynamic Web page generation request, said system comprising: . . .

Claims 9 and 10 of the '554 patent disclose substantially the same limitations of claim 1 except that the claimed steps are performed on a network system. As shown in Figure 3 above, the Popp system operated on a network of computers including, at a minimum, a Web Client computer and the computers running everything else including the Web server, the page servers and the data sources. Shamos Decl. at ¶¶135-143. Claim 9 also requires a "page server processing means," a "first computer system including means for generating said request," and "a second computer system including means for receiving said request." The Popp system discloses these structures and requires them to operate. *Id.* Additionally, as discussed above with respect to claims 1 and 9, Popp discloses claim 10's limitations including an "interceptor" and a "dispatcher" running on the system's network. The interceptor is the HTTP server software that performs the act of intercepting, and the dispatcher is the software that runs the CGI interface program (*e.g.*, CGIMessenger 210). *Id.* at ¶¶144-146. Accordingly, Popp discloses all the limitations of claim 9 and 10.

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: . . .

Claim 11 of the '554 patent discloses substantially the same limitations of claim 1 except that the claimed steps are performed when stored instructions on a computer readable medium are executed by a computer system. Because the apparatus of Popp is a programmed system whose instructions are on or are downloaded from a machine readable medium (e.g., a hard disk or CD), Popp discloses all the limitations of this claim. *Id.* at ¶¶147-148.

Claims 2 and 16 of the '335 patent (including limitations of Claim 1)

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

limitations of these claims, which are substantially identical to that of claim 1. *Id.* at ¶¶149-152.

D. The Asserted Claims Are Invalid as Anticipated by Dienst

1. The Dienst Reference is Prior Art Under 35 U.S.C. § 102(a)

The Dienst reference is an "Implementation Reference Manual" that describes the architecture and implementation of Dienst: a protocol and server that provides distributed documents libraries over the World Wide Web. A46. The Dienst reference is a technical report that was published by the Computer Science Department of Cornell University on May 5, 1995. *See* A47 [Declaration of Carl Logoze] at ¶¶2-6. Accordingly, Dienst is prior art to the patents-in-suit under 35 U.S.C. § 102(a) because it was accessible to the public before epicRealm's earliest asserted invention date of August 1995.

2. Dienst Discloses Every Limitation of Every Asserted Claim

As demonstrated below with respect to representative claim 1 of the '554 patent, the Dienst reference anticipates all the asserted claims of the patents-in-suit.

> **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:**

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 at MAC000281; Shamos Decl. at ¶154. These results that are presented to the user are dynamic Web pages. *Id.*

> **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,**

As shown in Figure 4 below, Dienst discloses routing a dynamic Web page generation request from a Web server (the WWW Server) to a page server (a Dienst Server) via a dispatcher

(a CGI Stub connected to a Dienst Server).

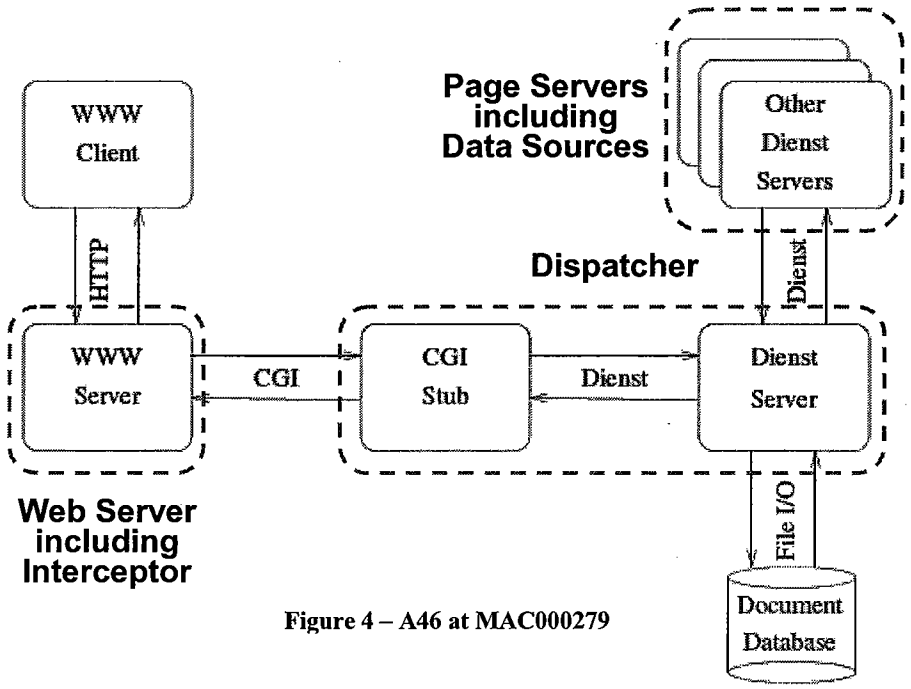


Figure 4 – A46 at MAC000279

The WWW server of Dienst is a "Web server" under both parties' proposed constructions because it is a server running HTTP-compliant software for receiving Web page requests and returning Web pages in response to those requests. Shamos Decl. at ¶155. Additionally, the Dienst server is a "page server" under epicRealm's proposed constructions because it includes software for generating dynamic Web pages:

> 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.

A46 at MAC000281. This "hit list" is a dynamic Web page because it is a "unified hypertext hit list" that is "presented to the user as a combined list of hyperlinks sorted by publisher and docid." *Id.*; Shamos Decl. at ¶156. Similarly, a Dienst server is a "page server" under Oracle's proposed construction for that term because it would be possible to run the server on a different machine

(*i.e.*, different processor) from that of the Web server over a "distributed network." *Id.*

Additionally, Dienst discloses "releasing said Web Server to process other requests" under epicRealm's application of both parties' proposed constructions for that claim term. First, under epicRealm's implicit releasing infringement theory, the Web server is "freed" to process requests by virtue of the fact that the page sever receives the request and relieves the Web Server of the obligation to process it. Shamos Decl. at ¶158. Additionally, under epicRealm's explicit releasing infringement theory, Dienst discloses several means by which the Dienst servers (the page servers) perform acts that, according to epicRealm, release the Web server to process other requests. *Id.* at ¶157. For example, the Dienst page servers release the Web server by sending TCP ACK messages and via Socket-Bind-Listen actions that, according to epicRealm's "TCP releasing" theory, frees up certain resources that would either allow processing of other requests or facilitate more efficient processing of already received requests. *Id.*

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;

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; Shamos Decl. at ¶159. 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. This act of distinguishing between static page requests and dynamic page requests and diverting the dynamic page requests away from the Web server to another program (*i.e.*, a page server) meets

both parties' proposed constructions for "intercepting." *Id.*

Additionally, Dienst discloses routing said request from said Web server (the WWW server) to a dispatcher. The Dienst CGI stub together with an interfacing Dienst server (*see* Figure 4 above) are a "dispatcher" and perform "dispatching" under both parties' proposed constructions for those terms. Shamos Decl. at ¶¶160-161. The CGI stub is an executable program that examines the incoming HTTP request, repackages the request, and dispatches it to an interfacing Dienst server. *Id.* at ¶160. The interfacing Dienst server can further "dispatch" requests to other Dienst servers ("index servers") that correspond to the publisher selections identified in the search request. A46 at MAC000281; Shamos Decl. at ¶161.

More precisely, the interfacing server examines the repackaged request to identify the publishers of the documents that the client is seeking. *Id.* The server then uses this information and dynamic information contained in a "meta-server" to select any other Dienst servers that can be used to process the request. *Id.* The request is then dispatched to those other Dienst servers. *Id.* The "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. . . . 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.

A46 at MAC000281.

Accordingly, the CGI stub and interfacing Dienst server dispatch requests according to epicRealm's proposed construction because they examine requests to make an informed selection of which page servers (*e.g.*, the "Other Dienst Servers" of Figure 4) should process requests based on dynamic information maintained about page servers. Shamos Decl. at ¶161. Because Oracle's proposed construction for "dispatching" is broader than that of epicRealm, these components must also perform dispatching under Oracle's proposed construction. *Id.*

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

Dienst discloses concurrency because it is inherent in the architecture shown above in Figure 4. *Id.* at ¶162. When the WWW server has passed on the request to a Dienst server, it may process other requests concurrently. *Id.*

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

As discussed above, Dienst discloses dynamically generating a Web page in response to the request which includes data dynamically retrieved from one or more data sources (*e.g.*, Dienst servers other than the interfacing Dienst server): "The user interface 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" is a Web page under both parties' constructions for that term, which can contain data (*e.g.*, search results) dynamically retrieved from one or more "Other Dienst Servers" (the data sources). Shamos Decl. at ¶163.

Claims 2-5 and 7-8 of the '554 Patent

Dienst discloses the step of claim 2 including identifying one or more data sources from which to retrieve said data. This occurs when a Dienst server examines environmental variables derived from the HTTP request to identify the appropriate data sources. *Id.* at ¶164. Dienst also discloses the step of claim 3 including dynamically retrieving said data from one or more data sources. *Id.* at ¶165. 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. *Id.* Although Dienst does not expressly disclose the steps of claims 4, 5, 7 and 8. It would be obvious to one of ordinary skill in the art to combine Dienst with the teachings of OWS 1.0 or 2.0 to meet the limitation of these claims. *Id.* at ¶¶166-169.

Claims 9, 10 and 11 of the '554 patent

Claims 9, 10 and 11 disclose substantially the same limitations of claim 1 except that the claimed steps are performed on a network system or when stored instructions on a computer readable medium are executed by a computer system. As shown in Figure 4 above, Dienst operated on a network of computers and was a programmed system whose instructions are on or are downloaded from a machine readable medium (e.g., a hard disk or CD). *Id.* at ¶¶170-183.

Claims 2 and 16 of the '335 patent (including limitations of Claim 1)

As discussed above with respect to claim 1 of the '554 patent, Dienst discloses all the limitations of these claims, which are substantially identical to that of claim 1. *Id.* at ¶¶184-186.

E. In the Alternative, the Asserted Claims Are Invalid Under § 103 as Obvious in View of Each Above Prior Art Reference in Combination with Garland

1. Garland is Prior Art under 35 U.S.C. § 102(b)

Another reference that discloses dispatching of Web page requests according to the patents-in-suit is Garland. A48. Garland on its face shows that it was published as a Carnegie Mellon University Technical Report on January 25, 1995 and was later published by the Defense Technical Information Center on March 20, 1995. *Id.* at ORCL01818177. Accordingly, Garland is prior art to the patents-in-suit under 35 U.S.C. § 102(b) because it was accessible to the public more than one year prior to the epicRealm patents' filing date of April 23, 1996.

2. Garland's Disclosure of "Dispatching" Can Be Combined with any of the Anticipation References as an Alternative Bases for Invalidity

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.

A48 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 web page requests by dispatching them to separate machines:

> 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 . . . .

Id. Garland further describes a particular distributed web server architecture where a ***dispatcher*** 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.

Garland is agnostic as to the type of processing done by the member servers, which can include dynamic Web page generation. Shamos Decl. at ¶189. Garland further discloses the use of dynamic load balancing to dispatch web page requests from the dispatcher server to the member servers according to dynamic load information exchanged between the dispatcher and the member servers. A48 at Section 3.4. Garland's "dispatcher server" performs "dispatching" according to both parties' proposed constructions because the dispatcher server makes informed selections of which member servers should process requests based on dynamic information (e.g., server load) maintained about the member servers. Shamos Decl. at ¶¶189-191.

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 its Introduction. *Id.* at ¶190. In other words, prior to epicRealm's earliest asserted invention date of August 1995, it was obvious to one of ordinary skill in the art to combine any Web server architecture with Garland's teaching of dispatching requests from a front-end server (*e.g.*, a web server) to two or more separate, distributed servers (*e.g.*, page servers) such that the load on the front-end server is reduced and such that the web server and the page servers can concurrently process requests. *Id.*

All of the asserted claims are anticipated by the above prior art references under epicRealm's application of the parties' claim constructions. The asserted claims are also obvious in view of OWS 1.0, OWS 2.0, Popp, or Dienst in combination with Garland. *Id.* at ¶¶191-192.

VI. CONCLUSION

For the reasons stated above, the Court should grant Oracle's Motion for Summary Judgment of Invalidity of the Asserted Claims.

MORRIS, NICHOLS, ARSHT & TUNNELL LLP

/s/ James W. Parrett, Jr.
Mary B. Graham (#2256)
James W. Parrett, Jr. (#4292)
1201 North Market Street
P.O. Box 1347
Wilmington, DE 19899
mgraham@mnat.com
jparrett@mnat.com
(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

Dated: July 31, 2008
2433311

CHART 1 - PARTIES' PROPOSED ELEMENT-BY-ELEMENT CLAIM CONSTRUCTION CHART

Claim Term	Oracle's Proposed Construction	epicRealm's Proposed Construction
connection cache ('554 patent – claim 4)	a store of information identifying open connections to data sources for eliminating subsequent connect times to those data sources	no construction is necessary
data sources ('554 patent – claims 1-5, 8-9, 11) ('335 patent – claims [1], [15])[1]	computerized repositories from which data may be retrieved electronically	no construction is necessary
dispatcher ('554 patent – claim 1, 9, 11) ('335 patent – claims 2, 16)	a software program for determining which page server should be used to process a dynamic web page generation request	no construction is necessary
dispatching / dispatching said request to said page server ('554 patent – claims 1, 9, 11) ('335 patent – claims 2, 16)	analyzing a request to make an informed selection of which page server should process the request, and sending the request to that page server	examining a request to make an informed selection of which page server should process the request based on dynamic information maintained about page servers, the dynamic information indicating which page server can more efficiently process the request, and sending the request to the selected page server
HTTP-compliant device ('335 patent – claims 15, 16)	a device running software that implements the Hypertext Transfer Protocol	a device that is compliant with the communication protocol known as HyperText Transport Protocol (HTTP)

[1] Claim numbers in brackets signify unasserted independent claims upon which asserted dependent claims rely.

Claim Term	Oracle's Proposed Construction	epicRealm's Proposed Construction
intercepting said request at said HTTP-compliant device ('335 patent – claim [1], [15])	receiving a request at the HTTP-compliant device and diverting the request before the HTTP-compliant device executable can process the request	at least intercepting the handling of a request at a said HTTP-compliant device
intercepting said request at said second computer system ('554 patent – claims 9, 10)	receiving a request at the second computer system and diverting the request before the second computer system executable can process the request	no construction is necessary
intercepting said request at said Web server ('554 patent – claims 1, 11) ('335 patent – claims [1])	receiving a request at the Web server machine and diverting the request before the Web server executable can process the request	intercepting the handling of a request at a Web server
interceptor ('554 patent – claim 10)	a software component that performs intercepting	no construction is necessary
logging into ('554 patent – claim 5)	presenting credentials that allow access to	no construction is necessary
machine readable medium ('554 patent – claim 11)	a hard disk, floppy disk, CD-ROM, magnetic tape, or other magnetic or optical data storage medium	no construction is necessary
page server ('554 patent – claims 1, 4, 7, 9, 10, 11) ('335 patent – claim [1], [15])	page generating software, running on a processor separate from that of the Web server, that generates dynamic Web pages Claims 9 of the '554 Patent and claims 15 and 29 of the '335 Patent are likewise construed with the substitution of "second computer system" and "HTTP-compliant device" respectively for "Web server."	page-generating software that generates a dynamic Web page

Claim Term	Oracle's Proposed Construction	epicRealm's Proposed Construction
releasing ('554 patent – claims 1, 9, 11) ('335 patent – claims [1], [15])	(see below)	freeing
said page server receiving said request and releasing said Web server [or second computer system or HTTP-compliant device] to process other requests ('554 patent – claims 1, 9, 11) ('335 patent – claims [1], [15])	said page server receiving said request and said page server performing an act (separate from merely receiving the request) to free the Web server to process other requests. Claims 9 of the '554 Patent and claims 15 and 29 of the '335 Patent are likewise construed with the substitution of "second computer system" and "HTTP-compliant device" respectively for "Web server."	(see proposed construction for "releasing" above)
request ('554 patent – all) ('335 patent – all)	a message that asks for a Web page specified by a URL	a message that asks for a Web page
router ('554 patent – claims 9, 10)	a device that intercepts and dispatches dynamic Web page generation requests	no construction is necessary
Web page ('554 patent – claims 1, 3, 6, 7, 9, 11) ('335 patent – claim [1])	a file containing embedded commands in a Web formatting language such as HTML, capable of being displayed on a Web browser	Web content displayable through a Web browser
Web server ('554 patent – claims 1, 11) ('335 patent – claim [1])	HTTP-compliant server software, or a machine running such software, that receives Web page requests and returns Web pages in response to the requests	software, or a machine having software, that receives Web page requests and returns Web pages in response to the requests

Claim Term	Oracle's Proposed Construction	epicRealm's Proposed Construction
Means for generating [said request] ('554 patent – claim 9)	§ 112 ¶6 corresponding function: generating said request structure: a processor of a computer that is, or has, a Web client running a Web browser	"a processor of a computer that is, or has, a Web client running a Web browser" or equivalents thereof
Means for receiving [said request from said first computer] ('554 patent – claim 9)	§ 112 ¶6 corresponding function: receiving said request from said first computer structure: a processor of a computer that is, or has, a Web server running a Web server executable	"a processor of a computer that is, or has, a Web server running Web server executable" or equivalents thereof
Page server processing means ('554 patent – claim 9)	§ 112 ¶6 corresponding function: processing dynamic Web page generation requests corresponding structure: a processor of a computer that runs software for generating dynamic Web pages	"a processor of a computer that runs page server software (wherein page server software is page-generating software that generates a dynamic Web page)" or equivalents thereof