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. AND RELATED COUNTERCLAIMS	C.A. No. 06-414 (SLR) **REDACTED PUBLIC VERSION**

DECLARATION OF DR. PAUL C. CLARK IN OPPOSITION TO EPICREALM'S MOTION FOR PARTIAL SUMMARY JUDGMENT OF LITERAL INFRINGEMENT

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
mgraham@mnat.com
jparrett@mnat.com

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

OF COUNSEL:

James G. Gilliland, Jr.
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: August 21, 2008

Redacted Filing: August 28, 2008

I, Dr. Paul C. Clark, declare:

1. I have been retained as a consultant and expert by Oracle in connection with the *Oracle Corporation and Oracle U.S.A. Inc. v. epicRealm Licensing, LP* matter. I have personal knowledge and am familiar with the matters described in, and the documents attached to, this declaration, which is submitted in support of Oracle's Opposition to EpicRealm's Motion for Partial Summary Judgment of Literal Infringement. If called to testify as a witness, I would be able to testify as to the statements and opinions set forth in this declaration. I have reviewed the numbered exhibits referenced in this declaration. I understand that these exhibits will not be attached to this declaration, but instead will be submitted in a separate appendix to be filed with the Court.

2. I was asked to offer an opinion as to whether Oracle's Internet Application Server product, including its optional Web Cache software, and Oracle's Database product, infringe certain claims of epicRealm's U.S. Patent Nos. 5,894,554 (the '554 patent) and 6,415,335 (the '335 patent). On June 6, 2008, I submitted a rebuttal report on noninfringement. This declaration is consistent with the analysis and opinions in that report, although this declaration contains only a subset of the analysis and opinions in the report. This declaration contains my analysis and opinions as to the accused Oracle products with respect to certain elements of epicRealm's asserted claims which are lacking in the accused products. I also provide opinions on the level of skill of one of ordinary skill in the art, and how one of ordinary skill in the art would have understood the '554 and '335 patents. This declaration contains my opinions to date. The basis for my opinions, the materials I considered in reaching them and my qualifications for rendering them are also set forth.

I. QUALIFICATIONS

3. Attached as Exhibit A to this declaration is a copy of my curriculum vitae, which more fully sets forth my background and qualifications.

4. In 1986, I received a Bachelor of Science degree in Mathematics from the University of California Irvine. In 1988, I received a Master of Science degree in Electrical Engineering and Computer Science from the University of Southern California. In 1994, I received a Doctor of Science degree in Computer Science from George Washington University.

5. From 1985 to 1989, I worked as a Systems Engineer at Ultrasystems Defense and Space. At Ultrasystems, I designed and implemented large-scale simulation and network-based systems for the United States Department of Defense (DOD). A custom high-speed database server I designed and implemented was used for real-time intelligence collection by the National Security Agency (NSA).

6. From 1989 to September 1990, I worked as a Technical Lead at GTE Government Systems. While at GTE, I designed and implemented network load generators for OS/2 LAN Manager to measure network performance load metrics for the Central Intelligence Agency (CIA). I also developed X Windows interfaces for a large-scale event driven network database system.

7. From 1990 to 1995, I worked as a Senior Security Engineer at Trusted Information Systems. While at Trusted Information Systems, I designed and implemented high assurance security systems, including trusted operating systems and applications for NSA and the Defense Advanced Research Projects Agency (DARPA). My work at Trusted Information Systems involved cryptography, multilevel operating systems, smartcards, and other cutting edge technologies.

8. From 1995 to 1999, I worked as Chief Scientist at DynCorp Network Solutions, where I served as senior internal security consultant for a variety of projects. For example, I was architect and Technical Director of the IRS Secure Submission and Retrieval System that allowed the digitally signed and encrypted submission of tax data over the Internet. The successful deployment of this system resulted in three Al Gore Hammer Awards. I also created a suite of security products for providing secure wide area access to database and application servers that was marketed and sold to the DOD and other parts of the federal government.

9. From 1999 to December 2007, I was the President and Chief Technology Officer of SecureMethods Inc. SecureMethods specializes in the design, implementation, and deployment of advanced secure network applications for commercial and government clients, including the DOD. In my capacity as President and Chief Technology Officer of SecureMethods, I had technical and operational oversight of all projects and corporate technical operations. My current work continues to include network systems and secure database applications. I am presently the President of Paul C. Clark LLC.

10. I was also a member of the Federal Advisory Committee for Key Management Infrastructure (KMI), serving as Chairman of the Interoperability Working Group for Cryptographic Key Recovery. I have also served as an adjunct professor in the Computer Science Department at the George Washington University, where I have taught doctoral-level computer security courses. I have also appeared before a Congressional committee to provide expert testimony on "Advanced Technology for Border Control."

11. I have co-authored a number of publications in the computer and security areas, and am a named inventor on two patents, U.S. Patent Nos. 5,448,045 and 5,892,902.

12. This experience has contributed significantly to my understanding of computer

networking technology, including the technology at issue in this litigation.

II. COMPENSATION TO BE PAID FOR WORK IN THIS CASE

13. I am being compensated at my customary rate of $400 per hour for technical consultation and testimony and immediate preparations for testimony. My compensation does not depend on the outcome of this litigation. In addition, I will be reimbursed for reasonable expenses incurred in connection with these activities.

III. MATERIALS AND INFORMATION CONSIDERED

14. In reaching my opinions, I have reviewed, among other documents: (1) the '554 and '335 patents and their prosecution file histories, (2) the infringement report of epicRealm's expert, Dr. David Finkel, (3) epicRealm's Motion for Partial Summary Judgment of Literal Infringement, (4) Dr. Finkel's declaration in support of epicRealm's motions, (5) the computer source code that defines the relevant structures and functions of Oracle's accused products, including the source code relevant to establish the presence or absence of the "intercepting," "dispatcher," "dispatching," and "releasing" limitations of the asserted claims, (6) the depositions of the inventors of the epicRealm patents, (7) the depositions of Oracle's engineers, and (8) various discovery responses and pleadings related to claim construction in this case. A complete listing of the documents I have reviewed and considered is attached hereto as Exhibit B. I have also drawn extensively on my own experience concerning computer network technologies.

IV. THE ASSERTED CLAIMS AND THE PATENTED TECHNOLOGY

15. I understand that epicRealm has accused Oracle of infringing claims 1-5 and 7-11 of the '554 patent and claims 2 and 16 of the '335 patent (the "Asserted Claims"). I further understand that epicRealm has accused (1) Oracle Web Cache, (2) Oracle Internet Application Server, and (3) Oracle Database (the "Accused Products") of infringing the Asserted Claims as

follows:

Accused Oracle Products	Releases at Issue	'554 Patent Claims Infringed	'335 Patent Claims Infringed
Accused Oracle Web Cache Products	Release 1.0.2 (November 2000) and all subsequent releases	1-5, 7-11	2 and 16
Accused Oracle Application Server Products	10gR1 (9.0.4) (April 2003) and all subsequent releases	1-5, 7-11	2 and 16
Accused Oracle Database Products	for RAC and JDBC: 10gR2 (10.2.0.1.0) (May 2005) and all subsequent releases; for RAC and OCI: 11g (11.1) (October 2007) and all subsequent releases	1-3, 7-11	2 and 16

16. I further understand that the meanings of certain claim limitations of the Asserted Claims are in dispute. Although the Court has not yet construed any disputed claim terms, both parties have proposed constructions for certain terms of the Asserted Claims, as set forth in the Joint Claim Construction Statement and related papers filed in this case. I have reviewed those proposed claim constructions and, where appropriate, have provided alternative non-infringement opinions for both parties' proposed constructions. Where Oracle has proposed a claim construction for a particular term and epicRealm has not, I have applied Oracle's proposed construction as part of my analysis discussed below, but I also have considered the claim language itself. I further understand that claim construction is a matter of law and that it is ultimately the Court's duty to construe the disputed claim terms. To the extent the Court adopts claim constructions that were neither proposed by Oracle nor epicRealm in this case, I reserve the right to offer additional opinions in view of those new constructions.

17. The patents-in-suit share a common specification and relate to structures and methods for managing dynamic Web page requests. Particularly, the claimed inventions focus

on the steps of: (1) intercepting a dynamic Web page request from a client at a Web server, (2) routing the request to a dispatcher, (3) dispatching the request to a page server, (4) the page server releasing the Web server to concurrently process other requests, and (5) the page server dynamically generating a Web page in response to the request based on data obtained from a data source. These steps are performed as shown in Figure 4 of the patents:

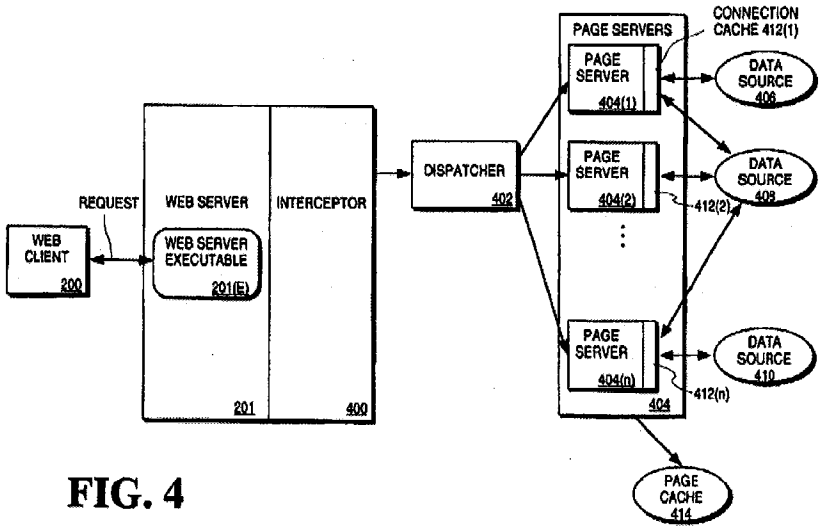


FIG. 4

18. The patent specification describes the prior art Web server environment as lacking "any mechanism for managing the Web requests or the Web sites." The patent further states that "[a]lthough these [Web server] machines may be running 'multi-threaded' operating systems that allow transactions to be processed by independent 'threads,' all the threads are nevertheless on a single machine, sharing a processor." The specification points out that "[w]hen numerous requests are being simultaneously processed by multiple threads on a single machine, the Web server can slow down significantly and become highly inefficient."

19. The claimed invention seeks to address this perceived inefficiency by "utilizing a partitioned architecture." This partitioned architecture has several components, including a Web

server (Fig. 4, item 201), an "Interceptor" (Fig. 4, item 400), a "Dispatcher" (Fig. 4, item 402), and one or more "page servers" (Fig. 4, items 404(1)-(n)).

20. Claim 1 of the '554 patent describes the method in which the components of this partitioned architecture are used to address the perceived inefficiency. The claimed invention describes the routing of requests from the Web server through a dispatcher to one of the page servers. The page server that receives the request releases the Web server to process other requests. After it has released the Web server, the page server processes the request, while the Web server that was released concurrently processes other requests.

21. The claimed invention defines how the Web server and page server communicate over the network. The Interceptor and Dispatcher are used to intercept a dynamic Web page request from a client at a Web server, and offload the request from the Web server by sending the request to a dispatcher and dispatching the request to a page server.

22. The claims require a separate Dispatcher, to which requests are routed from the Web server. The patent provides that the Dispatcher program may be placed on the same machine as the Web server, but that the Dispatcher program must be capable of being off-loaded onto a separate machine as the Web site grows, which is only possible if it is a separate software program from the Web server program. A separate Dispatcher program is also required for the claimed step of "routing the request [from a Web server] to a Dispatcher." This claimed limitation cannot be satisfied unless the Dispatcher is a separate program from the Web server program, because a software program cannot "route" to itself—a software program can only "route" data to a separate program—so the Dispatcher program must be a separate executable program from the Web server program and capable of running on a separate machine.

23. The claims do not explicitly require a separate Interceptor, but all the claims at

least require that the act of "intercepting" be done at the Web server.

24. The Interceptor intercepts the request at the Web server and routes it to the Dispatcher instead of the Web server executable processing the request. The Dispatcher receives the intercepted request and then dispatches it to one of the page servers. When the page server receives the request from the Dispatcher, the page server releases the Web server, so that it may concurrently process other requests. After releasing the Web server, the page server begins processing the request, while the Web server is concurrently processing other requests.

25. The page server processes the request in its possession by dynamically generating a Web page, in response to the request, based on data obtained from a data source. The claimed invention is used in a system that includes one or more "data sources" (Fig. 4, items 406, 408, 410, 414). Data sources "include databases, spreadsheets, files and any other type of data repository."

26. By using the page servers to release the Web server to concurrently process other requests, the claimed invention seeks to address the perceived inefficiency of the prior art Web server that uses multiple processing "threads" on a single machine.

V. MY ANALYSIS OF THE ACCUSED PRODUCTS

A. Oracle's Accused Web Cache Product

1. What is Web Cache

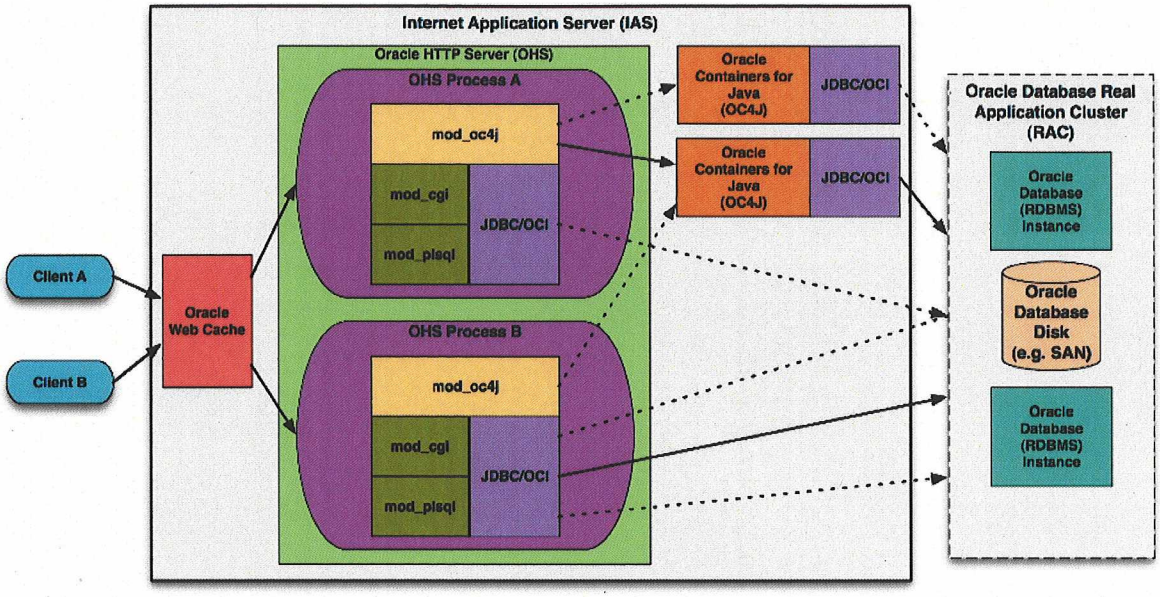
27. The optional Web Cache component of Oracle's Internet Application Server product is a software program known as a cache server. A cache server is used to reduce the burden on a Web server. When confronted with increasing demands on a Web server, a Website operator could reduce the burden by adding additional Web servers, but the computer hardware and management costs associated with adding more Web servers often outweighs the benefits.

Instead, an operator may install a cache server, such as Web Cache, in front of the Web server.

28. Web Cache, like all cache servers, is designed to maintain a cache, or a local store, of frequently accessed Web page URLs in its memory. A "URL" or Universal Resource Locator, is a unique identifier, or address, that defines the location of a file on the Web or any other Internet facility. An example would be "http://www.oracle.com". Web Cache sits in front of the Web server, and receives the Web client's request for content before the Web server does. If Web Cache has the requested content in its cache (a "cache hit") then the Web Cache simply returns the requested content to the Web client, eliminating the need to repeatedly process identical requests through the application Web server and database. Only if the requested content is not cached (a "cache miss") does Web Cache send the request to a Web server for processing. In the Web Cache source code and documentation, the Web servers sitting behind Web Cache that originate new content in the event of a cache miss are called "origin servers."

29. Web Cache is able to cache both static and dynamic content. Algorithms within the Web Cache will periodically flush dynamic content from the cache, which will force the next request for that dynamic content to be serviced by the origin server again. Web Cache does this in order to avoid sending outdated dynamic content to the Web client. Caching of both static and dynamic content reduces the load on the origin server sitting behind the Web Cache.

30. Cache hits are handled completely by Web Cache, while Web Cache sends cache misses to the origin server and waits for them to be processed. Web Cache can be configured in many different ways to fit the needs of the user. For a very high-traffic Website, the Website operator can configure multiple Web Cache servers to sit in front of the origin servers, to provide failover protection and load-balancing.



31. In a typical configuration of Internet Application Server in which the Web Cache option is installed (see above illustration), a single Web Cache process sits in front of a single origin server. Users can, however, also configure a system to use a single Web Cache in front of multiple origin servers, where the Web Cache maintains a cache of content requested from each origin server, and distributes cache misses among the multiple origin servers. However, Oracle's documentation discourages use of this configuration for high-traffic Websites, because using Web Cache as a type of software load-balancer establishes the Web Cache server as an inefficient single point of failure in the system. In this configuration, if the Web Cache server malfunctions under the high load, then the entire system is disabled and the websites become unavailable. Instead, the systems used as examples in Oracle's Web Cache documentation typically use third-party hardware load balancers to distribute requests, and leaves Web Cache to perform the function it was designed for: caching, as reflected in the figure above.

32. Regardless of how Web Cache is configured, its purpose is the same: to cache frequently requested content in order to reduce the load on the origin server. Web Cache is

designed so that its caching functionality is performed within a single multi-threaded program, wherein each Web request is delegated to a "fiber" within the Web Cache program. Each fiber is an independent unit of execution, and the resources provided to each fiber are managed by the Web Cache program. Web Cache is further divided into a "Front End" that manages fibers that connect to Web clients and perform cache lookups, and a "Back End" that manages fibers that connect to the origin server and add new content to the cache.

33. Front End fibers are created on-the-fly upon receipt of a request, up to a user configured maximum number of fibers. When the Web Cache process receives a request from a Web client, the Web Cache process first creates a new Front End fiber to handle that request. The default maximum number of Front End fibers is 700, which means that Web Cache can support up to 700 simultaneous Web client requests in its default configuration. If all 700 fibers are occupied—either processing a request or waiting for a requested Web page from an origin server—then Web Cache cannot process any additional requests until one of the 700 fibers completes a request.

34. After creating a new Front End fiber to handle the request, that Web Cache fiber compares the URL of the request to the URLs of prior requested content that it has stored within its cache. In the event of a cache hit, the Front End fiber returns the requested content to the Web client, processing of the request is completed, and the Web Cache process destroys the Front End fiber. Web Cache does not normally keep the Front End fiber alive to handle subsequent requests.

35. In the event of a cache miss, then the Web Cache process will create a Back End fiber that connects to the origin server. The Back End fiber communicates the URL of the request to the origin server, and then waits for the origin server to process the request and return

the content. The Front End and Back End fibers will continue to wait until either the origin server returns the requested content, or a time-out error occurs.

36. Once the origin server returns the requested content, the Front End and Back End fibers return the content to the Web client and insert the content into the cache so that future requests for the same content can be satisfied by the Web Cache without involving the origin server. This accomplishes Web Cache's intended purpose to reduce the load on the origin server. Once these steps are completed and the Web client has received the requested content, the Web Cache process destroys the Front End and Back End fibers. Fibers are created and destroyed on-the-fly and are not used to process additional requests.

37. As mentioned above, Web Cache does not normally keep the Front End fiber alive to handle a second request; new Front End fibers are created and destroyed on a per-request basis. The only exception is if the Web client sends further requests over the same connection by means of a feature of version 1.1 of the HTTP protocol known as the `Keep-Alive` command. Then the Front End fiber will process each of the additional requests, one at a time, in the same fashion I describe here, before being destroyed. Regardless of whether `Keep-Alive` is used, a fiber can handle only one request at a time, and is never free to concurrently process another request when waiting for requested content from an origin server.

2. Web Cache Neither Has an Interceptor Nor Performs Intercepting

38. EpicRealm contends that the term "intercepting" means "intercepting the handling of a request at a Web server." EpicRealm's construction contains a circular definition that uses the term "intercepting." Oracle contends that the term "intercepting" means "receiving a request at the Web server and diverting the request before the Web server executable can process the request." I understand both parties' proposed constructions to require the "diversion" or

"interception" of a request to a dispatcher before the Web server "processes" or "handles" the request.

39. Dr. Finkel describes "intercepting" under epicRealm's proposed construction as follows:

> Logic associated with Oracle Web Cache instructs the intercepting of a request and routes it to a dispatcher. The intercepting occurs when the determination is made that the "Web server" (for example, Oracle Web Cache) will not satisfy the request itself with pre-existing or static content, but rather, the request will be satisfied with dynamic content by a "page server" (for example, Oracle Application Server including Oracle HTTP Server).

A27, Exh. E, at p. 6 (emphasis mine). Dr. Finkel's description of epicRealm's theory does not materially differ under Oracle's proposed construction. A27, Exh. E, at p. 7.

40. I agree with Dr. Finkel's factual observations that, during normal operation of the Web Cache software, certain requests will be processed solely by Web Cache (cache hits) and certain requests will be processed both by Web Cache and by OHS (cache misses). However, I disagree with Dr. Finkel's theory that Web Cache performs the "intercepting" limitation of the asserted claims. Again, I don't perceive there to be any factual dispute between Dr. Finkel and me over how Web Cache actually operates.

41. Web Cache does not "intercept" because it does not make the determination to satisfy a request from "pre-existing," or static, content, or to not satisfy the request itself because it calls for dynamic content. The Web Cache does not distinguish between static and dynamic content when it is processing a request. Web Cache stores both static and dynamic content, and the function of the Web Cache is to determine if the requested content, whether static or dynamic, is within its cache. Indeed, requests for static pages not found in the cache are still "processed" or "handled" and then sent to an origin server. Thus, Web Cache's processing does

not involve a determination based upon a "static vs. dynamic" distinction, or any other distinction, and thus fails to meet the "intercepting" limitation of the asserted claims.

42. Web Cache also does not "intercept" because the determination that "the 'Web server' (for example, Oracle Web Cache) will not satisfy the request itself" is not made until after the Web Cache has completed processing or "handling." As I stated above, I understand epicRealm's construction to require the "interception" of a request to occur before the Web server "handles" the request.

43. I have reviewed the source code relevant to how Web Cache "processes" or "handles" a request. First, Web Cache receives a request and creates a new Front End fiber to handle the request. Next, Web Cache checks its cache for the requested content. If the requested Web page is found in the cache (a cache hit), Web Cache returns the Web page to the client and processing of the request is completed. If the requested Web page is not found (a cache miss), Web Cache creates a new Back End fiber to connect to OHS and transmit the request for further processing. Web Cache will then wait until OHS returns the requested Web page. When OHS returns the Web page, Web Cache inserts a copy of the page into its cache in order to service future requests for the same page, and returns the Web page to the client, completing the request. These processing steps are followed for every request made by every client, irrespective of the type of Web content called for by the Web page request.

44. My review of this source code supports my conclusion that Web Cache will send a request to an origin server only after it has completed its processing of the request. Since intercepting must occur before the Web Cache (the accused "Web server") "processes" or "handles" the request under both parties' proposed claim construction it is clear that Web Cache does not "intercept." Thus, the Oracle Web Cache product does not perform the "intercepting" of

the patent.

45. Therefore, it is my opinion that dynamic Web requests are not "intercepted" as required by the asserted claims under either epicRealm's or Oracle's proposed constructions.

3. Web Cache Has No Dispatcher

46. All of the Asserted Claims require a Web server routing to a "dispatcher" for "dispatching" requests from a Web server to a "page server." I understand that epicRealm contends that no construction of "dispatcher" is necessary, while Oracle contends that a "dispatcher" is "a software program for determining which page server should be used to process a dynamic Web page generation request." I understand that, at this time, neither party has proposed a construction for the term "routing."

47. A person having ordinary skill in the relevant art of computer science would understand the ordinary meaning of the term "routing" to mean "the directed transmission of data packets over a computer network." Packets of data can be routed from one computer to another over a network, such as Ethernet. Packets of data may also be routed between two software programs on the same computer using a network interface called the loopback. However, the source and destination must be different software programs capable of running on separate computers. In other words, a software program cannot "route" data to itself. I believe that this is evident even if one uses a more generic definition of routing, such as to send from one place to another.

48. This understanding is consistent with the specification of the patents-in-suit. The specification discloses two embodiments of the Dispatcher of the claimed inventions. According to the specification, both embodiments of the Dispatcher must be able to reside on a separate computer from the Web server. Col. 5, 8-10, 31-34. This also is consistent with epicRealm's

statements to the Patent Office during prosecution of the '335 patent. In response to a rejection of claims, epicRealm differentiated its invention from prior art U.S. Patent No. 5,752,246 ("Rogers") and U.S. Patent No. 5,701,463 ("Malcolm") by representing to the examiner that the cited prior art "does not teach or suggest 'routing a request from a Web server to a page server' because mere interception and determination do not teach or suggest 'routing' the request *from one place to another*." A4, at p. EPIC000550. (emphasis added). Therefore, based upon my general knowledge of the relevant art, and my analysis of the patents-in-suit and their file histories, it is my opinion that the claimed "dispatcher" must be contained in a separate software program from the claimed "Web server."

49. EpicRealm's infringement contentions, and the report of its expert, Dr. Finkel, identify Web Cache as both the "Web server" and the "dispatcher" of the asserted claims. For epicRealm's theory to be correct, Web Cache must be able to route intercepted requests to itself.

50. The relevant functionality of Web Cache – its caching function and its communication of Web requests to origin servers – is entirely contained within a single software program. An example of a familiar software program is Microsoft Word. Microsoft Word is a single executable program, contained in a file, "word.exe". The Word program has different features and functions, such as the print function, the print preview function, and the file open function. These functions are all contained within the same software program. Web Cache is structured the same way – it contains different features and functions, but they are all part of the same software program. Web Cache is not designed to use a separate dispatcher, and was likely designed this way to avoid the additional overhead and use of resources that would necessarily accompany the use of a separate dispatcher program.

51. Because Web Cache is a single software program, it is incapable of "routing" to

itself, and therefore cannot be both the "Web server" and the "dispatcher" under epicRealm's infringement theory. Therefore, it is my opinion that Web Cache does not infringe the asserted claims because it lacks a "dispatcher."

4. Web Cache Is Not "Released" by a "Page Server"

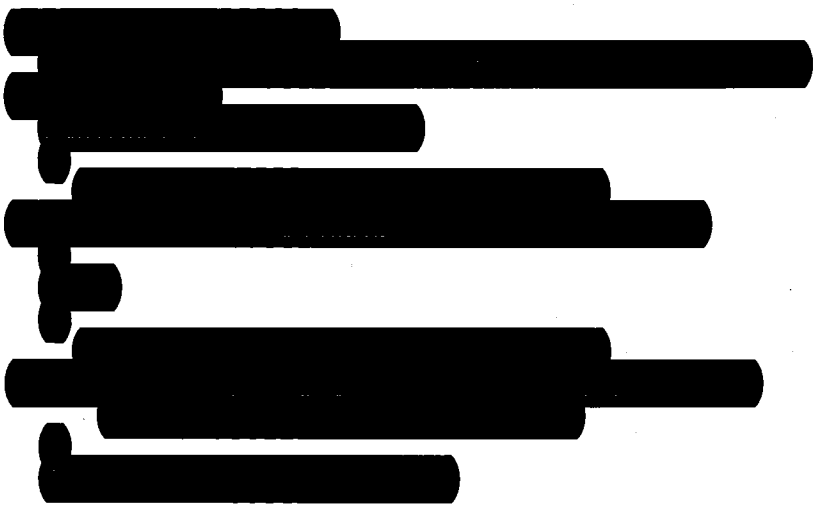
52. All of the Asserted Claims require that the page server, upon receipt of a request, "release" the Web server to concurrently process other requests. EpicRealm contends that the term "releasing" means "freeing." Oracle contends that "said page server receiving said request and releasing said Web server to process other requests" means "said page server receiving said request and said page server performing an act (separate from merely receiving the request) that expressly communicates to said Web server that is may now resume processing other requests."

53. Dr. Finkel states in his report that Web Cache is the accused "Web server" of claims 1-5 and 7-11 of the '554 patent and claim 2 of the '335 patent (or the accused second computer system of claims 9-10 of the '554 patent, or the HTTP-compliant device of claim 16 of the '335 patent). He further identifies "Oracle Application Server including Oracle HTTP Server" as the accused page server of claims 1-5 and 7-11 of the '554 patent and claim 2 of the '335 patent. Dr. Finkel also opines that Web Cache is "released" in one of two ways: (1) Web Cache is implicitly released when the request is sent from Web Cache to the Oracle Application Server including Oracle HTTP Server, or (2) Web Cache is explicitly released when the Oracle Application Server including Oracle HTTP Server, upon receiving the request, affirmatively communicates a message to Web Cache that releases Web Cache to concurrently process other requests.

54. I agree with Dr. Finkel's factual observations that, during normal operation of the Web Cache software, requests are sent from the Web Cache to OHS (in the event of a cache

miss). However, I disagree with Dr. Finkel's conclusion that Web Cache is "released" to process other requests when an Oracle Application Server including Oracle HTTP Server is processing a dynamic Web request.

55. I have reviewed the relevant source code sections relating to how Web Cache behaves when sending a request to an origin server:



56. This source code establishes that a Web Cache Back End fiber (and its corresponding Front End fiber) tasked with a request *always waits* until the origin server returns the requested page, or until a time-out occurs.

57. As the source code demonstrates, the Web Cache fiber assigned to handle a given request is unavailable to handle another request until it returns the completed first request (or an error message) to the Web client. Moreover, if the Web client has not used the HTTP `Keep-Alive` command, then the fiber never processes another request because it is destroyed by the Web Cache process after completing one request. The fact that the Web Cache process may create other fibers to handle additional requests is not relevant to the "releasing" claim limitation, because those Web Cache fibers were never tasked with the first Web request to begin with, and thus never "released" in any way when the first Web Cache fiber sent its request to an origin

server and began waiting.

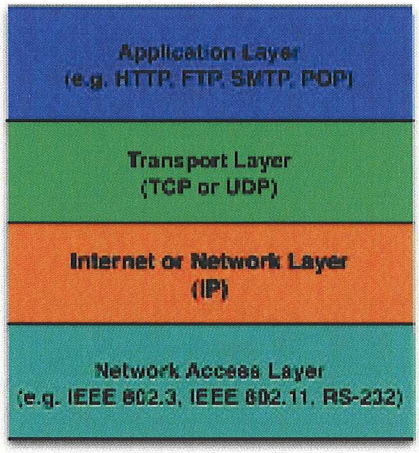
58. I agree with Dr. Finkel's factual observations that, during normal operation of the Web Cache software, TCP acknowledgements are sent from the computer running OHS to the computer running Web Cache. However, all TCP/IP communications from the alleged page server of Figure 4 of the patent would necessarily be with the dispatcher, not the Web server. Thus, I disagree with Dr. Finkel's opinion that the Web Cache is "released" to process other requests by means of a TCP acknowledgement sent from an Oracle Application Server including Oracle HTTP Server. I have analyzed the source code and other documentation that describes how Web Cache utilizes the operating system's TCP/IP stack to communicate to an origin server (e.g. an Oracle HTTP Server). This analysis confirmed my conclusion that Web Cache is not "released" to process other requests by means of an acknowledgement over TCP/IP. I do not believe that there is any factual dispute between Dr. Finkel and myself as to how Web Cache is operating.

59. The Internet protocol suite (commonly TCP/IP) is the set of communications protocols that implement the protocol stack on which the Internet and most commercial networks run. It is named for two of the most important protocols in it: the Transmission Control Protocol (TCP) and the Internet Protocol (IP).

60. The Internet protocol suite can be viewed as a set of layers. Each layer solves a set of problems involving the transmission of data, and provides a well-defined service to the upper layer protocols based on using services from some lower layers. Upper layers are logically closer to the user and deal with more abstract data, relying on lower layer protocols to translate data into forms that can eventually be physically transmitted. The TCP/IP reference model consists of four layers. From lowest to highest, these are the network access layer, the network

layer, the transport layer, and the application layer.

Figure 2: The Internet Suite of Protocols (The "TCP/IP" Model)



61. The layers of the Internet protocol suite are defined as follows:

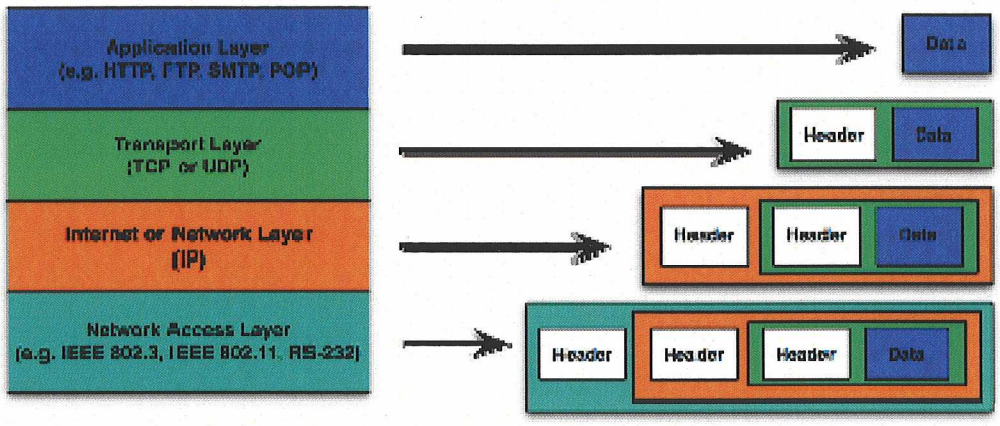
- Application Layer: The application layer is best described as all of the processes that involve user interaction. The applications within the layer determine the presentation of the data and control the session. An example of an application would be a Web browser such as Microsoft Internet Explorer or Mozilla Firefox, which primarily use Hypertext Transfer Protocol (HTTP) as an application layer protocol to communicate with servers running Websites. Software applications that use HTTP, such as a Web browser, are not the only computer programs that reside on the application layer. An application is any process that occurs above the transport layer. There are numerous application layer protocols in the Internet protocol suite, including Simple Mail Transfer Protocol (SMTP) and Post Office Protocol (POP) used for e-mail, and File Transfer Protocol (FTP) for the transfer of files to and from various computers.
- Transport Layer: In the Internet protocol suite, there are two transport layer protocols: The Transmission Control Protocol (TCP) and the User Datagram Protocol

(UDP). TCP is a "reliable" transport protocol, because it guarantees that information is received as it was sent. UDP, in contrast, performs no such reliability guarantees.

- Network Layer: The Network Layer isolates the upper layer protocols from the details of the underlying network and manages the connections across the network. The Internet Protocol (IP) is normally described as the TCP/IP network layer. All upper and lower layer communications travel through IP as they are passed through the TCP/IP protocol stack.
- Network Access Layer: The network access layer is a combination of the data-link layer protocols that define the use of electrical signals for data transmission, and the physical layer protocols that define the physical connection itself. For the Internet and modern commercial networks, the most common network access layer protocol is IEEE 802.3, which describes the Ethernet standard for data-link and physical layer standards.

62. The four layer structure of TCP/IP is built as information is passed down from applications to the network access layer. When data is sent, each layer treats all of the information it receives from the layer above as data, and adds control information to the front of that data. This control information is called a header, and the addition of a header is called encapsulation. When data is received, the opposite procedure takes place as each layer removes its header before passing the data to the layer above.

Figure 3: Encapsulation of Data with Header Information at Each Protocol Layer



63. The above illustration depicts the standard TCP/IP communication model. A software program operating at the application layer, such as a Web server (using the HTTP protocol) would send a communication by invoking the TCP transport layer present in the computer operating system. The TCP layer would encapsulate the HTTP communication with a TCP header that contains the necessary information to deliver the data via TCP. The transport layer would then send the communication packet to the IP protocol, which again encapsulates the data with its own header, and so forth.

64. The receiving computer would work in the opposite fashion: The IP layer would use the IP header information and then discard it, sending the TCP-encapsulated data to the transport layer, where TCP would use the TCP header information, discard it, and then send the HTTP communication to the Web application on the receiving end.

65. This layered model of computer communication means that computer software can be designed and written using a modular, "object-oriented" approach. A software developer designing Web software using an application protocol such as HTTP does not have to concern him or herself with the details of implementing TCP or other lower layer protocols. Those

details are already taken care of by other software, such as the computer operating system (*e.g.*, Microsoft Windows, Sun Solaris, GNU/Linux), and the application layer is ignorant of the inner workings of the lower layers.

66. During TCP communication between two computers, the operating system of the receiving computer will transmit a transport layer acknowledgement (ACK) message when it receives a TCP segment. If the ACK for that TCP segment is not received by the operating system of the sending computer within a certain time, the operating system of the sending computer will resend that TCP segment. Once this ACK is received by the operating system of the sending computer, it then discards the received packets from the TCP memory buffer.

67. Importantly, the Oracle accused products do not send or receive TCP ACK messages. As explained above, the TCP layer is implemented by the computer's operating system software, and is transparent to application layer software such as the Oracle accused products. The operating system software sends and receives TCP ACK messages, and the Oracle accused products are oblivious to their transmission or receipt.

68. I agree with Dr. Finkel's factual observations that ACK messages received by the sending computer serve to free small amounts of buffer memory allocated by the computer operating system. However, Dr. Finkel's report asserts that the freed memory buffer space here satisfies the "releasing" limitation of the asserted claims. This is wrong. Dr. Finkel's alleged "explicit releasing" of a TCP memory buffer affects operations of the transport layer, but it has no effect on, and is transparent to, the application layer where the Web server functionality resides.

69. The Web Cache is not freed to process additional requests by virtue of the underlying operating system deleting TCP segments from the TCP memory buffer in response to

a received ACK message. EpicRealm has not identified a "page server" (page generating software) for this theory—they have only identified "IAS." IAS is the name of an Oracle product that is comprised of several software programs, it is not a software program itself. The page generating software is either OHS or OC4J, however, neither OHS not OC4J sends ACK messages—as discussed above, only operating systems send ACK messages. Even if an operating system or systems is considered to be part of the undefined "page server," ACK messages from OC4J's operating system cannot free Web Cache because they are sent to OHS's operating system, not Web Cache's operating system. An ACK message sent by OHS's operating system to Web Cache's operating system could not be considered an ACK to a "Web server" because this would ignore the requirement in the claims and the specification that the "page server" communicate with the "dispatcher," not the "Web server." Additionally, ACK messages sent from OHS to Web Cache do not release Web Cache to process other requests. As discussed above for "implicit releasing," the Web Cache source code establishes that the Web Cache fibers wait when they send requests to OHS—they are not freed to process other requests. Even if "releasing resources" such as operating system memory were considered to be a way that a Web server could be released, Dr. Finkel cannot show that an ACK from a "page server" is causing Web Cache's operating system memory to be released, or that any such memory is being allocated to the Web Cache, as opposed to other programs on the Web Cache machine.

70. If an operating system is waiting to execute a TCP send because its TCP buffer is full, a received ACK that clears space in that TCP buffer does not permit that fiber to concurrently process another request, it merely permits the operating system to finish executing its TCP send and begin waiting for the origin server to send back the requested content.

71. Further, the receiving of an ACK message, or failure to receive one, does not

affect the operation of other Web Cache fibers. In the accused Web Cache product, the fibers do not share TCP connections. Each fiber is allocated its own TCP buffer. Thus, if a Web Cache fiber is waiting to execute a TCP send because its TCP buffer is full, whether or not an ACK is ever received does not affect the ability of other Web Cache fibers to receive and process other requests. Again, each Web Cache fiber is independent of the other, and thus there can be no "releasing" as required by the asserted claims.

72. As evidenced by the Web Cache source code in ¶55 above, a Web Cache fiber, when processing a Web request, will wait after making the "receive" function call. The fiber will stop waiting when it receives the requested content back from the origin server that was processing the request. Any number of ACK messages can be sent and received while the Web Cache fiber is waiting in this manner, and none of these ACK messages will permit this fiber to stop waiting and begin processing other requests.

73. Ultimately, a Web Cache fiber is able to begin processing another request only when the first request is completed (by either success or error) and the Web client has used the HTTP `Keep-Alive` command. Thus a Web Cache fiber is never able to concurrently process additional requests as required by the "releasing" limitation of the asserted claims.

74. All of the claims in the '554 and '335 patents that have been asserted by epicRealm in this lawsuit contain this "releasing" limitation. As explained above, the accused Oracle Internet Application Server product does not "release" (implicitly or explicitly) Web Cache under either epicRealm's or Oracle's proposed construction of the releasing limitation. Therefore, I have concluded that Oracle's Web Cache product does not infringe the asserted claims.

B. Oracle's Accused Internet Application Server Product

1. What is Internet Application Server

75. Oracle's Internet Application Server ("IAS") is a product that includes multiple software programs, such as Oracle HTTP Server ("OHS") and Oracle Containers For Java ("OC4J"). OHS is the Web server component of Oracle Application Server, and consists of the HTTP Listener and a collection of modules, which run within a single software executable. An "executable" (for a single-threaded program such as Oracle HTTP Server) is a single instance of a computer program (a series of instructions that carry out an algorithmic solution to a task) that is executed sequentially by a computer system. OHS is based on the Apache Web server, a Web server software product developed by the open-source software community and distributed for free over the Internet.

76. HTTP Listener: Oracle HTTP Server is based on an Apache HTTP listener to serve client requests. An HTTP server listener receives incoming requests and passes them to the appropriate processing module.

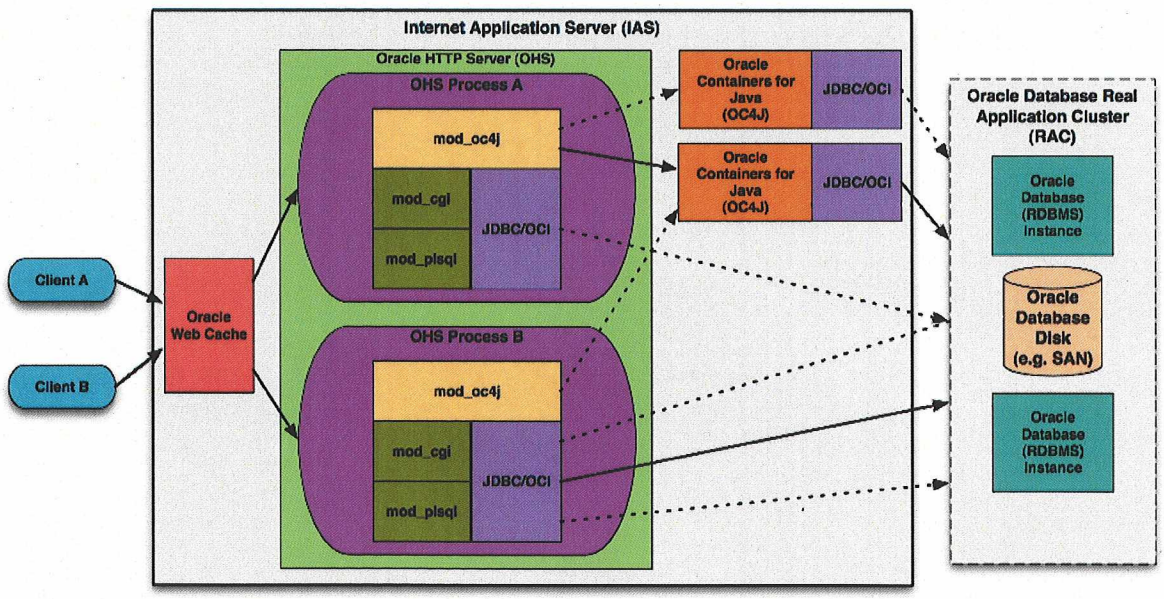
77. Modules (mods): Modules that perform various functions related to the processing of Web page requests. Many of the standard Apache modules are included with Oracle HTTP Server. Oracle also includes several modules that are specific to Oracle Application Server.

78. OHS modules extend the basic functionality of the Web server, and support integration between Oracle HTTP Server and other Oracle Application Server components. For example, mod_perl permits the Oracle HTTP Server to process requests that call for the execution of a software application written in the Perl programming language. OHS is capable of including numerous mods. While much of OHS is based on Apache, Oracle adds its own

mods to provide support for certain functionality proprietary to Oracle. One of those is mod_oc4j; a module that enables OHS to communicate with OC4J.

79. Mod_oc4j handles Web requests that require processing by a Java software application. Mod_oc4j provides the OHS executable program with the ability to route requests from the Oracle HTTP Server to an OC4J program. OHS, using mod_oc4j, communicates with OC4J through an application-level protocol called the Apache Java Protocol (AJP) version 1.3.

80. OC4J is a separate executable program within Oracle's Internet Application Server product, and is designed to contain an Oracle customer's Java-based software applications. An Oracle customer using OC4J would design a Java-based software application, and then use OC4J to run that application when it is requested by a Web client. OHS provides access for Web clients to access the customer's Java-based application from the Web.



81. The illustration depicts how a Web client's request for a Java-based Web page gets processed. First, the Web client sends a request to run a Java application to the IP address of the computer running Oracle HTTP Server. If the optional Web Cache is present, and the

Web page is in the cache, Web Cache returns the Web page and the request is completed. If Web Cache is not present in the Web site configuration, or if the Web page is not found in the cache, then that request is accepted by one of the instances (i.e., copies) of the OHS executable running on that computer. In the default configuration, there are 256 instances of OHS running at any given time. Once a particular instance of OHS has received the request, that OHS executable processes the request to completion.

82. OHS begins processing a request by calling each module, in the order in which they were loaded into memory when the OHS program was first started. In addition to the mod_oc4j and mod_plsql modules developed by Oracle, IAS includes many modules that are developed and maintained by third-parties within the open-source community, such as mod_cgi, mod_fastcgi, mod_perl, and mod_php. When it calls a module, OHS compares the URL of the request against a list of Web page content types to determine whether it is the type of request that the module is designed to process. Once OHS identifies the correct module to use, OHS uses that module to process the request.

83. If the OHS process gets through calling all of the modules and no module has the capability to process the request, the OHS process will attempt to satisfy the request from the static or generated dynamic content available to it from the computer's local storage (the file system on the computer's hard drive). If there is no file in local storage that satisfies the request, then OHS will send an error message to the Web client that the requested content was not found. When the OHS process sends such an error message, the OHS process performs no further processing of the request. Only when the OHS process sends either the requested Web page content or an error message to the Web client is the OHS process done processing the request.

84. In the case of requests for Web content using a Java-based application, mod_oc4j

is the module that would handle the request. The OHS process would execute the mod_oc4j function. Upon execution, the OHS process, using mod_oc4j, would send the request to an OC4J process, and then wait until OC4J returned the completed request. OHS is not designed to execute Java applications on its own. OC4J is the program in the Oracle software that generates Web pages in response to Web page requests for Java-based content (OC4J is not used to process requests for non-Java Web page content). When OC4J completes its processing of the request and sends the requested Web content back to the OHS process, the OHS process stops waiting, and finishes processing the request by sending the requested content back to the Web client and then waits for another request.

85. If another Web client has sent a second request when this particular OHS process is waiting, that second request is accepted by one of the other OHS processes.

86. Users of Oracle's Internet Application Server product can change some of the configuration details concerning how mod_oc4j communicates with OC4J processes. OC4J processes (also called "instances") are started and managed by Oracle Process Manager and Notification Server (OPMN). Users must configure OHS with mod_oc4j to operate. *See* A39, at pp. ORCL00372158 – 168. When more than one OC4J process exists in a user's IAS configuration, mod_oc4j will load balance requests across multiple instances of OC4J application servers. Users can configure mod_oc4j to perform one of eight different load balancing policies:

Table 1: Mod_oc4j Load Balancing Policies (A39, at p. ORCL00372164)

Name	Description
Random	`mod_oc4j` randomly selects an OC4J instance from a list of OC4J instances that are candidates to service a request.
Round Robin	`mod_oc4j` randomly selects an OC4J instance from an ordered list of OC4J

Name	Description
	instances that are candidates to service a request. Other OC4J instances are selected from the ordered list in turn, until the initially selected server is selected again. This sequence is repeated. If a particular OC4J instance is stopped or is unavailable, then that instance is skipped (no attempt is made to select it) until it can be brought back in service.
Random with Local Affinity	`mod_oc4j` randomly selects local OC4J processes to service requests. When no local OC4J processes are available, `mod_oc4j` randomly selects remote OC4J processes and gives them equal opportunity to be selected.
Round Robin with Local Affinity	`mod_oc4j` routes all requests to local OC4J processes in a round robin manner. When no local processes are available, `mod_oc4j` routes requests equally to each OC4J process on different hosts.
Random using Routing Weight	`mod_oc4j` distributes requests according to the routing weight configured for each host. One OC4J process is selected randomly from the OC4J processes on that host.
Round Robin using Routing Weight	`mod_oc4j` distributes the total request load to OC4J processes on each host based on the routing weight configured to each host. `mod_oc4j` selects an OC4J process in round robin manner from the OC4J processes on that host.
Metrics Based	`mod_oc4j` routes requests as per the run time metrics from OC4J processes that indicate how much load is currently being placed on the OC4J process.
Metric Based with Local Affinity	`mod_oc4j` routes all requests to local OC4J processes as per the run time performance metrics of OC4J processes. When there are no local OC4J processes available, `mod_oc4j` routes requests to each OC4J process on different hosts as per their performance metrics only.

87. The default load balancing configuration for mod_oc4j is Round Robin. Two of the eight different load balancing policies enable mod_oc4j to route requests based on metrics from OC4J processes that, for example, indicate how much load is currently being placed on the OC4J process. Those two metrics-based policies are "Metrics Based" and "Metric Based with Local Affinity" as described in the chart above. If a customer wishes to use one of these metrics-based load balancing policies, the customer must change the configuration of mod_oc4j; otherwise the default Round Robin load balancing is used.

88. Based upon my personal knowledge and experience in the field, I agree with Dr. Finkel, that metrics-based load balancing—distributing requests for processing across multiple

computer machines—was a well known concept and practice in computer science at the time of the claimed invention. *See* A74 [Finkel Dep.], at 31:24-34:11, 178:14-181:5.

89. Round Robin is a form of load balancing based on a simple sequential algorithm. Mod_oc4j maintains a list of the OC4J processes, and will send the first request to the first OC4J process on the list, the second request to the second process, and so forth. If the next OC4J process on the list to handle a request is still busy processing the prior request assigned to it, then mod_oc4j will skip over that OC4J process and send the request to the next available OC4J process on the list. This Round Robin method is the default method used by OHS processes to determine which OC4J process should receive a given Web request for Java-based content.

2. IAS Neither Has an Interceptor Nor Performs Intercepting

90. EpicRealm contends that the term "intercepting" means "intercepting the handling of a request at a Web server." EpicRealm's proposed construction contains a circular definition that uses the term "intercepting." Oracle contends that the term "intercepting" means "receiving a request at the Web server and diverting the request before the Web server executable can process the request." I understand both parties' proposed constructions to require the "interception" or "diversion" of a request to a dispatcher before the Web server "processes" or "handles" the request.

91. Dr. Finkel describes "intercepting" under epicRealm's proposed construction as follows:

> Logic associated with Oracle HTTP Server instructs the intercepting of a request and routes it to a dispatcher. The intercepting occurs when the determination is made that the "Web server" (for example, Oracle HTTP Server) will not satisfy the request itself with pre-existing or static content, but rather, the request will be satisfied with dynamic content by a "page server" (for example, an OC4J process).

A27, Exh. C, at p. 6 (emphasis mine). Dr. Finkel's description of epicRealm's theory does not materially differ under Oracle's proposed construction. A27, Exh. C, at p. 9.

92. Oracle HTTP Server does not "intercept" because it does not make the determination either to satisfy a request from "pre-existing," or static, content or to not satisfy the request itself because it calls for dynamic content. The OHS process does not make a determination of how to satisfy a request based upon a "static or dynamic" distinction. The determination of which module is suited to handle a request is pre-determined, based upon the URL of the request and the configuration file of the module. For example, a request for Perl-based content will be handled by mod_perl regardless of whether the requested content is static or dynamic. The Perl script, when executed by the OHS process using mod_perl, may return static content, such as plain text or an image file, or it may return dynamic content, such as the contents of an online shopping cart. Thus Oracle HTTP Server's "processing" or "handling" of a request is not based on any "diversion" of requests for dynamic Web pages, as required by the intercepting limitation, such as by using a "static vs. dynamic" distinction, or any distinction.

93. Oracle HTTP Server also does not "intercept" because the determination that "the 'Web server' (for example, Oracle HTTP Server) will not satisfy the request itself" is not made until after the OHS program has "processed" or "handled" the request. As I stated above, I understand epicRealm's construction to require the "interception" of a request to a dispatcher before the Web server "handles" the request.

94. I have reviewed the source code relevant to how Oracle HTTP Server "processes" or "handles" a request. When an OHS process receives a request, it immediately begins processing the request by calling each module. Each module will handle a request that it is configured to handle. For example, if the requested page contains Java-based content, mod_oc4j

will handle that part of the request. If the requested page contains a Perl script, mod_perl will handle the processing of the Perl program. When OHS goes through the iterative process of calling each of the module functions available to it to process a request, OHS is processing the request. This calling of each module to process requests is occurring within the OHS program itself, and is the primary function of OHS's processing capability.

95. Only after the OHS process gets through calling all of the modules and no module has the capability to process the request, will the OHS process attempt to satisfy the request from the static or dynamic content available to it on the local computer. If there is no local file that satisfies the request, then OHS will send an error message to the Web client that the requested content was not found. When OHS sends such an error message, it performs no further processing of the request. When OHS sends either the requested Web page content or an error message to the Web client, OHS is done processing the request. This process is performed by each instance of OHS for each request.

96. There is no intercepting at any step of OHS's processing of a request. There is no intercepting when OHS receives a request from a Web client because the request is still at the Web server executable. Likewise, there is no intercepting of the request when OHS invokes its various modules, because the modules are part of the OHS executable program as well, and when OHS invokes its modules it makes no distinction between dynamic and static requests.

97. Thus, the Oracle HTTP Server does not perform the "intercepting" required by the patent.

3. IAS Has No Dispatcher

98. All of the Asserted Claims require a "dispatcher" for "dispatching" requests from a Web server to a "page server." For the reasons explained in ¶¶46-48 above, the patents require

that the "dispatcher" must be a separate software program from the "Web server."

99. EpicRealm's infringement contentions, and the report of its expert, Dr. Finkel, identify Oracle HTTP Server the "Web server" and mod_oc4j as the "dispatcher" of the asserted claims. For epicRealm's theory to be correct, Oracle HTTP Server must be able to route intercepted requests to mod_oc4j.

100. My review of the source code established that mod_oc4j is entirely contained within the OHS program. Each OHS process has its own copy of mod_oc4j (and all other installed modules) within that executable. Because OHS and mod_oc4j are contained within a single software program, Oracle HTTP Server is incapable of "routing" to mod_oc4j, and therefore mod_oc4j cannot be the "dispatcher" under epicRealm's infringement theory. OHS is not designed to use a separate dispatcher, and was likely designed this way to avoid the additional overhead and use of resources that would necessarily accompany the use of a separate dispatcher program.

101. Therefore, it is my opinion that the Internet Application Server product does not infringe the asserted claims because it lacks a "dispatcher."

4. IAS Does Not Have a Page Server that "Releases" a Web Server

Implicit Releasing

102. All of the Asserted Claims require that the page server, upon receipt of a request, "release" the Web server to concurrently process other requests. EpicRealm contends that the term "releasing" means "freeing." Oracle contends that "said page server receiving said request and releasing said Web server to process other requests" means "said page server receiving said request and said page server performing an act (separate from merely receiving the request) that expressly communicates to said Web server that it may now resume processing other requests."

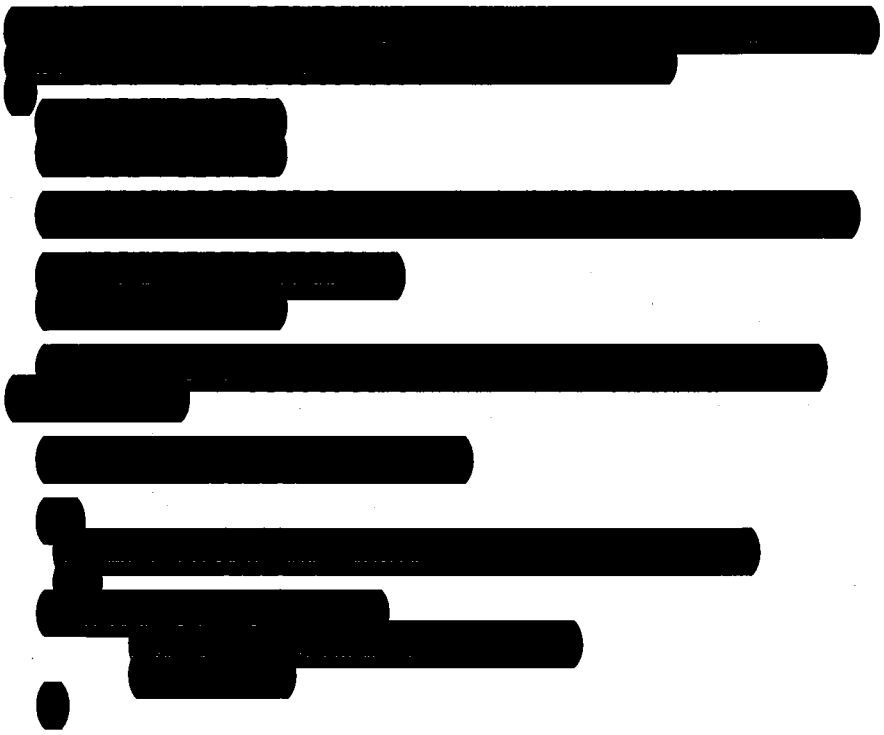
103. Dr. Finkel states in his report that the Oracle HTTP Server contained within Oracle's IAS product is the accused "Web server" of claims 1-5 and 7-11 of the '554 patent and claim 2 of the '335 patent (or the accused second computer system of claims 9-10 of the '554 patent, or the HTTP-compliant device of claim 16 of the '335 patent). He further identifies "an OC4J process" as the accused page server of claims 1-5 and 7-11 of the '554 patent and claim 2 of the '335 patent. Dr. Finkel also opines that the OHS is "released" in one of two ways: (1) the Web server is implicitly released when the request is sent from the Web server to the page server, or (2) the Web server is explicitly released when the page server, upon receiving the request, affirmatively communicates a message to the Web server that releases the Web server to concurrently process other requests.

104. I agree with Dr. Finkel's factual observations that, during normal operation of the OHS software, requests are sent from OHS to OC4J (in the event of a request to execute a Java application), and TCP acknowledgements are received by the computer running OHS from the computer running OC4J (and vice versa). However, I disagree with Dr. Finkel's conclusion that OHS is implicitly or explicitly "released" as set forth in either of Dr. Finkel's proposed theories. Again, I don't perceive there to be any factual dispute between Dr. Finkel and me over how OHS actually operates.

105. As I explained in my description of Oracle's Internet Application Server product in ¶¶81-85 above, an OHS process, once it receives a request from a Web client, is not able to receive or process any other requests until that request is completed, either by the satisfaction of that request or by the return of an error message to the Web client that made the request.

106. I have reviewed the relevant source code sections relating to how an OHS process behaves when sending a request to an OC4J process. This portion of source code establishes that

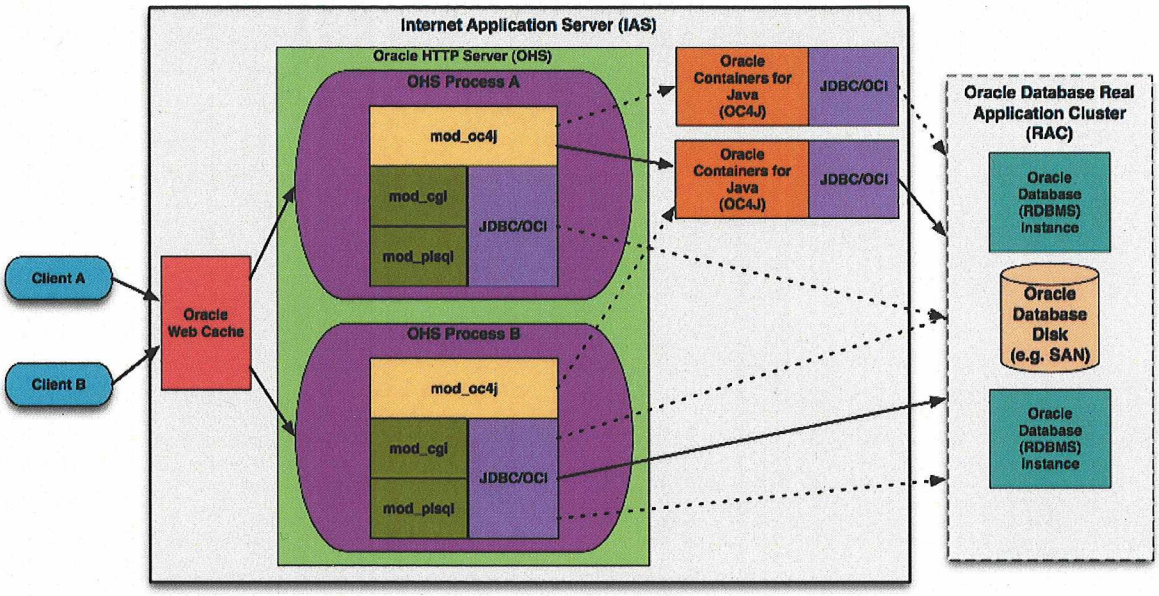
the OHS process waits until the OC4J process returns the requested page.



107. The claimed inventions require that the released Web server be able to concurrently process other requests as a result of the page server releasing it to do so, while the page server is processing the first request. This required element of the claim is not present in the accused Oracle Internet Application Server product. As the source code demonstrates, the instance of OHS assigned to handle a given request is unavailable to handle another request until it returns the completed request or an error message to the Web client. If another Web client sends a request, that request is handled by one of the other available instances of OHS. The fact that there may be other instances of OHS available to handle additional requests is not relevant to the "releasing" claim limitation, because those instances of OHS were never tasked with the first Web request to begin with, and thus never "released" in any way when the first instance of OHS sent its request to an OC4J process and began waiting. If, hypothetically, all of the OHS

processes were busy processing requests or waiting for requested Web pages from OC4J processes, then the Web server would be unable to process any additional requests until one of the OHS processes completed its request.

108. The diagram illustrates how Oracle's Internet Application Server product operates to handle Web page requests:



109. In the illustration, Client A sends Request A to the Internet Application Server product. Let us assume, for this example, Web Cache does not have the requested Web page (if it did, Web Cache would return the Web page and Client A's request would be complete). Request A is received by OHS process A. OHS process A then begins processing the request by calling each of the module functions so that those module functions can determine whether the capability exists to satisfy the request. In this illustration, the request is for a Web page that contains a Java-based application, so OHS process A uses mod_oc4j to send the request to an OC4J process for further processing. Once OHS process A sends the request to OC4J, OHS process A blocks, and is incapable of receiving or processing another request.

110. Now, Client B initiates a dynamic Web page request. Again, let us assume Web Cache does not have the requested Web page. Because OHS process A is blocked, it cannot receive Request B. OHS process B is available because it is not involved at all with the processing of Request A, so it receives Request B and begins processing, again by calling the various mods to determine how to satisfy the request. If additional clients sent Requests C, D, etc., those requests would be received and processed by different instances (i.e., copies) of the OHS executable in the same fashion.

111. The above example illustrates that, while OHS process B is capable of concurrently processing Request B while one of the OC4J processes is processing Request A, what OHS process B does is irrelevant. OHS process B is not involved in any step of processing Request A and therefore can never be "released" by the OC4J process that has received Request A. It is OHS process A that must be "released . . . to process other requests" and it clearly is not.

112. Dr. Finkel's employer-employee analogy in his declaration does not accurately illustrate how OHS and OC4J operate. The employer in Dr. Finkel's analogy is never freed by the employee to perform other tasks. When the employee (OC4J) is processing a request for an OHS process (the employer), that OHS process waits, unable to process other requests until OC4J returns the requested Web page. If OC4J is not installed on the computer system and OHS receives a request for a Java-based Web page, then OHS will ultimately return an error to the Web client because, as discussed above, OHS is unable to process Java requests on its own. In the absence of the employee, the employer would be unable to complete the "duty, task, or obligation" of satisfying the request. Hence, I disagree with the employer-employee analogy.

113. All of the claims in the '554 and '335 patents that have been asserted by epicRealm in this lawsuit contain this "releasing" limitation. As I have explained here, the

accused Oracle Internet Application Server product does not "implicitly release" a Web server as required by epicRealm's proposed claim construction. Therefore, I have concluded that Oracle's Internet Application Server product does not infringe the asserted claims.

Explicit Releasing

114. IAS uses TCP/IP as its "Reliable Network Protocol." TCP/IP is not located in the accused Oracle Internet Application Server product; the code for TCP/IP is located in the computer operating system software used by the customer's machine. I have analyzed the source code and other documentation that describes how OHS utilizes the operating system's TCP/IP stack to communicate to an OC4J process. This analysis confirmed my conclusion that no Web server is released by means of a TCP acknowledgement. Further, as shown in Figure 4 of the patent, all TCP/IP communications must necessarily occur between the page server and the dispatcher not the page server and the Web server.

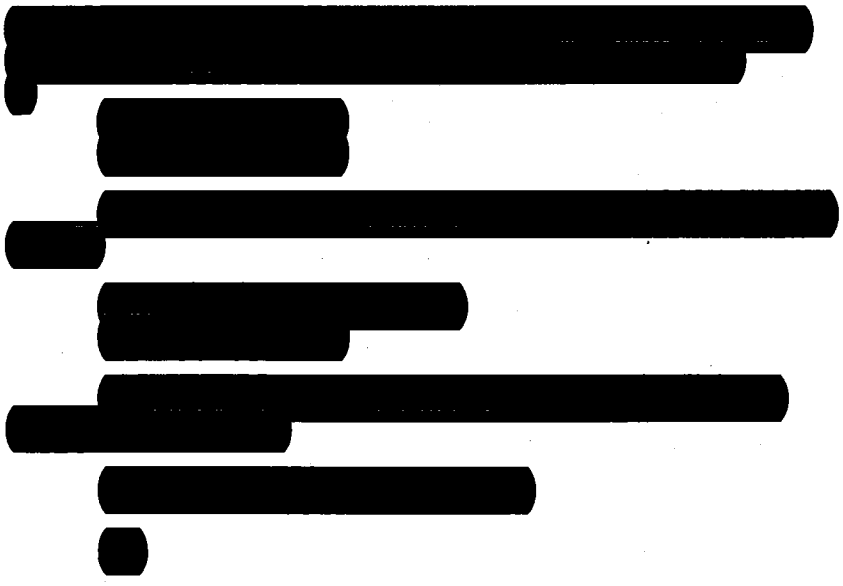
115. I have provided a general discussion of TCP in ¶¶59-66, above. The Oracle HTTP Server is not freed to process additional requests by virtue of the operating system deleting TCP segments from the TCP memory buffer in response to a received ACK. As I have explained above in my discussion of "implicit" releasing, the OHS process handling a given request will always wait until the OC4J process has completed the request. This remains true irrespective of any network "acknowledgement" that may be sent by the receiving computer running the OC4J process. In other words, even if the operating system underlying the OHS process receives an ACK, the OHS process continues to wait.

116. If an OHS process is waiting to execute a TCP send because its TCP buffer is full, a received ACK that clears space in that TCP buffer does not permit that OHS process to concurrently process another request; it merely permits the operating system to finish executing

its TCP send and begin waiting for the OC4J process to send back the requested content. The only things that would allow the OHS process to process another request are: (1) when the first request is completed, or (2) the system "times out" and delivers an error back to the Web client. In neither of these two situations can the OHS process concurrently process another request while waiting for OC4J to complete its processing of the first request, which is the "releasing" that the asserted claims require.

117. Further, the receiving of an ACK message, or failure to receive one, does not affect the operation of other OHS processes. In the accused Oracle IAS product, the OHS processes do not share TCP connections. Each OHS process is allocated its own TCP buffer. Thus, if an operating system is waiting to execute a TCP send because its TCP buffer is full, whether or not an ACK is ever received does not affect the ability of other OHS processes to receive and process other requests. Again, each OHS process is independent of the other, and thus there can be no "releasing" as required by the asserted claims.

118. My conclusion that no "explicit" releasing occurs is supported by the same source code from Oracle HTTP Server that I cited in my discussion of "implicit" releasing:



[redacted]

119. When processing a Web request, will wait upon making the [redacted] in the above source code. The OHS process will only stop waiting when it receives the requested content back from the OC4J process that was processing the request, or if a time-out occurs. Any number of ACK messages can be sent and received while the OHS process is waiting in this manner, and none of these ACK messages will permit the OHS process to stop waiting and begin processing other requests.

120. I disagree with epicRealm's premise that TCP ACK messages can serve as the "releasing" act required by claim 11 and the other asserted claims. The '554 patent could not possibly contemplate ACK messages sent by the operating system of a "page server" to constitute "releasing" of the "Web server." Those ACK messages are being sent as a result of the dispatcher software program, not the Web server software program, dispatching a request to the page server. The dispatcher's receipt of the Web page request from the Web server would cause the operating system on which the dispatcher is running to send an ACK message to the Web server, regardless of whether the request is ever successfully dispatched to the "page server." If the sending of an ACK message could be considered "releasing" of the Web server, it would be the operating system of the dispatcher, not the operating system of the page server, that would be sending the "releasing."

121. The '554 patent explicitly teaches against epicRealm's theory that an ACK message generated by the "page server's" operating system in response to receipt of a request for

a dynamic Web page is in any way "releasing" the Web server. The specification states that the preferred embodiment of the claimed invention places the dispatcher software program on a machine separate from that of the Web server. The separate operating system of this stand-alone dispatcher would send ACK messages as a result of the dispatcher's receipt of the request from the Web server. A "page server's" receipt of the request from the dispatcher, if it happens at all, would send an ACK to the dispatcher's operating system, not the Web server's. The resources "freed," if any, would be on the dispatcher's operating system. The preferred embodiment would not be covered by the claims because the ACK message from the page server's operating system to the dispatcher's operating system would not "release" the Web server or its operating system."

122. If the "page server" includes the operating system, this would not cover the alternative embodiment in which the dispatcher program resides on the same machine as the Web server. The TCP message would be sent by the "page server" operating system as a result of the receipt of request from the dispatcher software program, not from the Web server program (executable).

123. Additionally, the memory in a TCP buffer that is freed as a result of an ACK is not necessarily, or even usually, allocated to other OHS processes. The operating system may allocate the memory to the execution of other programs running on the machine, such as firewalls, system management software, and other networking programs (e.g., Domain Name Server programs). Dr. Finkel has not shown why or how the memory freed by a TCP ACK would necessarily be allocated to another OHS process, or how it would have any other effect on the OHS Web server software. For these reasons, it is my opinion that TCP ACKs cannot be the "releasing" act required by the claims.

124. Ultimately, the OHS process is able to begin processing another request only

when the first request is completed (by either success or error), and thus the OHS process is never able to concurrently process additional requests as required by the "releasing" limitation of the asserted claims.

125. All of the claims in the '554 and '335 patents that have been asserted by epicRealm in this lawsuit contain this "releasing" limitation. As I have explained in this section, the accused Oracle Internet Application Server product does not "release" (implicitly or explicitly) a Web server under either epicRealm's or Oracle's proposed construction of the releasing limitation. Therefore, I have concluded that Oracle's Internet Application Server product does not infringe the asserted claims.

C. Oracle's Accused Database Product

1. What is Oracle Database

126. Oracle Database primarily consists of a Relational Database Management System ("RDBMS"). An RDMBS is a complex package of software programs that control the organization, storage, management, and retrieval of data in a database (e.g., a structured collection of records or data) based on a relational model. The primary role of an RDBMS is to service requests for data stored in the database. Those requests are typically made in the form of database queries, which are requests for data that are made in a language that the database management system can understand (e.g., the Structured Query Language ("SQL")).

127. It is my understanding that Oracle customers use Oracle Database to organize, store, manage, and retrieve data, not Web pages. Although the data retrieved from an Oracle Database may be displayed in a Web browser, such Web pages are typically constructed by a separate application that is connected to the Oracle Database (e.g., OC4J or mod_plsql) or by a client's browser.

128. In the Oracle Internet Application Server architecture, for example, a client request that seeks Java content will get passed from the Oracle HTTP Server to an OC4J server using mod_oc4j. OC4J can interface with Oracle Database for the retrieval of raw data to be used in the construction of Web pages. An OC4J server running Java accesses the Oracle Database via an application programming interface called JDBC (see discussion in next section). Once the necessary data is retrieved from the Oracle Database, OC4J can construct a requested Web page using Java. Accordingly, in this common example, the Oracle Database is not used for the storage or generation of Web pages.

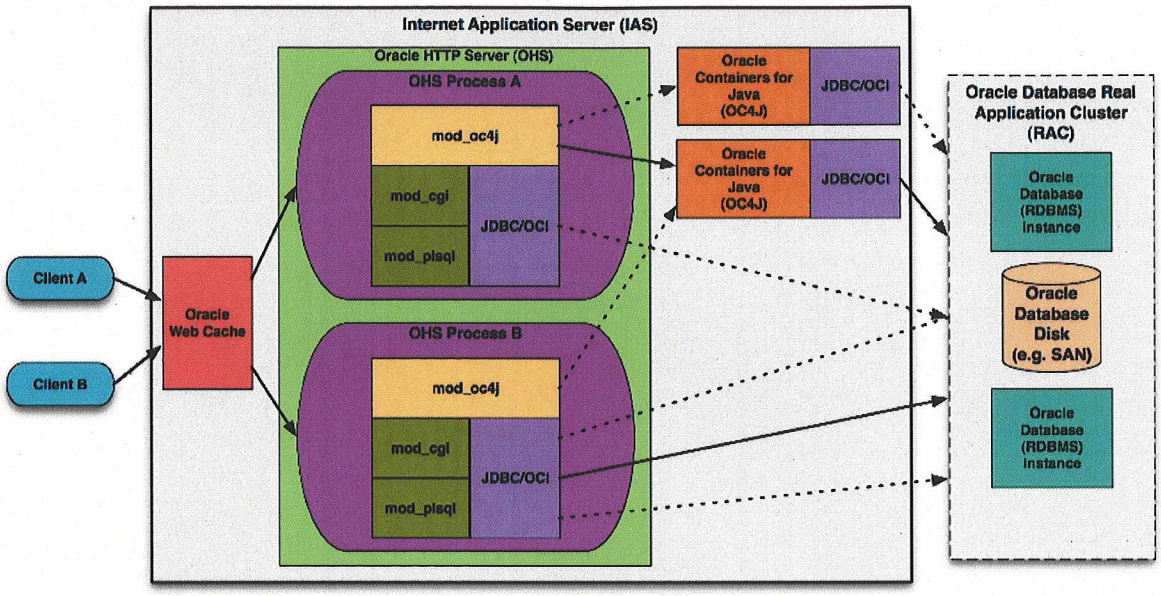
129. Application servers and other Database clients can connect to an Oracle Database in a variety of ways for the execution of database queries. One way an application program running on a server can connect to Oracle Database is via an application programming interface ("API"). An API, or Application Programming Interface, is not a separate executable program: it is a set of standardized function calls that permit a software program to issue high-level commands through a library that is linked into the program.

130. JDBC is an industry standard set of APIs that provide the interface for connecting from Java applications to relational databases, including Oracle Database, for the purpose of issuing database queries. There is one industry standard of APIs for JDBC. Database vendors, such as Oracle, can provide additional vendor-specific JDBC drivers that support database specific functionality. JDBC is used for accessing databases from a Java application.

131. Similarly, C and C++ applications can also connect to the Oracle Database using APIs called the Oracle Call Interface ("OCI") and the Oracle C++ Call Interface ("OCCI"), respectively. OCI, for example, consists of a set of C-language software APIs that provide a low-level interface to the Oracle Database. OCI offers a procedural API for not only performing

certain database administration tasks (such as system startup and shutdown), but also for using PL/SQL or SQL to query, access, and manipulate data. Here, both JDBC and OCI are libraries that permit the OHS program to issue commands to the Oracle Database. There is software code located on both the OHS and the Database that includes this library, but that code resides in the OHS and the Database executables, not in any separate program.

132. Oracle Real Application Clusters (RAC) is a database option in which a single database is accessed by multiple instances on multiple computers or "nodes." RAC provides for high availability and scalability of the Oracle Database. RAC's shared disk method of clustering access to databases increases scalability because nodes can easily be added or freed to meet current needs and improve availability because if one node fails, another can assume its workload. It adds high availability and failover capability to the database, because all instances have access to the whole database. A typical RAC implementation groups multiple Oracle database instances running on multiple nodes. These nodes communicate with each other and share a common pool of disks. These disks house all of the data files that comprise the database. This architecture is illustrated in the figure below:



2. Database Neither Has an Interceptor Nor Performs Intercepting

133. Infringement of all the Asserted Claims requires the "intercepting" of a request at a "Web server" (or a "Second Computer System" or a "HTTP-compliant device"). Dr. Finkel opines in his report that the accused "page server" of Oracle's Database product is a RAC instance. A27, Exh. G, at p.2. Dr. Finkel also opines in his report that the accused "intercepting," under both parties' proposed claim constructions, occurs at OHS (or at the computer running OHS). Dr. Finkel provides no other examples of an accused Web server (or a "Second Computer System" or a "HTTP-compliant device") that can be used with Oracle Database to allegedly infringe an Asserted Claim.

134. It is my opinion that Oracle's Accused Database Products cannot be used to infringe any Asserted Claim because requests are never intercepted at OHS. As in my discussion of Internet Application Server above in ¶¶90-95, OHS and the computer running OHS do not "intercept" requests under either party's proposed claim construction for that term.

3. Database Neither Has a Dispatcher Nor Performs Dispatching

135. All of the Asserted Claims require a "dispatcher" for "dispatching" requests from a Web server to a "page server." For the reasons explained in ¶¶46-48 above, the patents require that the "dispatcher" must be a separate software program from the "Web server."

136. EpicRealm's infringement contentions, and the report of its expert, Dr. Finkel, identify Oracle HTTP Server the "Web server" and JDBC and OCI as the "dispatcher" of the asserted claims. For epicRealm's theory to be correct, Oracle HTTP Server must be able to route intercepted requests to JDBC and OCI.

137. My review of the source code established that JDBC and OCI are not separate programs. JDBC and OCI are software libraries that are linked into the OHS executable when OHS is first started. Hence, the code and functionality of JDBC and OCI are entirely contained within the OHS program. Each instance of OHS has its own copy of the JDBC and OCI libraries. Because OHS, JDBC and OCI are contained within a single software program, OHS is incapable of "routing" to JDBC or OCI, and therefore JDBC and OCI cannot be the "dispatchers" under epicRealm's infringement theory. Therefore, it is my opinion that the Database product does not infringe the asserted claims because it lacks a "dispatcher."

138. Using the example of Microsoft Word as above in Paragraph 50, the Word program links in software libraries, such as printer drivers, that allow Word to communicate with a particular model of printer—however, the printer driver is not a stand-alone executable program; it merely allows software programs such as Word to use the printer. JDBC and OCI are similar to the printer driver, in that they are merely used by the software program into which they are linked (OHS and OC4J) to send queries to a database. There is no separate executable program, and thus no "dispatcher."

139. I understand that epicRealm has proposed a new theory of infringement, where the OC4J program, in conjunction with JDBC, is identified as the alleged dispatcher and the RAC database instance is the alleged page server. This theory is incorrect because, as discussed above in Paragraph 84, when OC4J is used (which is whenever there is a request for Java-based Web content), the requested Web page is generated in OC4J, not in the database—which means that the RAC Database instance cannot be the claimed page server under this theory. When OC4J sends queries to a database during its execution of a Java application, those queries are for the raw data that OC4J uses to generate the Web page, not Web pages themselves. In situations where the RAC Database instance may be generating a Web page, such as when the Database processes a PL/SQL stored procedure, OC4J is not involved at all—OC4J is only involved in the processing of Java-based Web page requests.

140. Claim 11 and the other asserted claims require "dispatching said request to said page server." OC4J, not the RAC Database instance, must be the "page server" under this new theory, because that is where page generation takes place for all Java-based Web page requests. To the extent that epicRealm asserts that OC4J is both the "dispatcher" and "page server," I disagree with epicRealm's new theory, because an OC4J program cannot "dispatch" requests to itself under either parties' proposed construction of "dispatching." OC4J with JDBC only sends requests to the database, never to itself.

141. Under both parties proposed constructions, Dr. Finkel opines that JDBC and OCI perform "dispatching" according the Asserted Claims by using the following requisite components: (1) an Oracle RAC Database having two or more instances that are configured to receive requests, (2) connection caching in the case of JDBC and session/connection pooling in the case of OCI, and (3) runtime load balancing. A84 [Finkel Infringement Report], Ex. G, at

63-67.

142. Under epicRealm's proposed claim construction for "dispatching," Dr. Finkel fails to identify how JDBC or OCI examines a request to make an informed selection of which page server should process the request based on dynamic information maintained about page servers. Assuming that "Runtime Load Balancing metrics" is "dynamic information maintained about pager servers," neither JDBC nor OCI examines a request to make an informed selection based on such dynamic information. I have reviewed the relevant source code sections relating to how JDBC and OCI send database queries to RAC instances and have confirmed that neither the JDBC nor OCI interface provide access to the Web request received by OHS, and thus they cannot examine the Web request to make an informed decision based upon runtime load balancing metrics.

143. For connections made to a RAC instance using JDBC and runtime load balancing, for example, JDBC is going to select a connection to a particular RAC instance (as determined by the RAC Load Balancing Advisory) using the `getConnection` function. The Oracle DriverManager interface defines three different forms of the `getConnection` method: `getConnection(String)`.

144. All of the information needed to describe the desired connection is encoded into the String parameter. `getConnection(String, Properties)`.

145. Some of the information is coded into the String parameter. The rest is passed as key value pairs in the Properties parameter.

146. The `getConnection(String, String, String)` method is a convenience method that passes the username and password as arguments rather than encoding them into the `Properties` parameter.

147. Additionally, the Oracle DataSource interface defines two `getConnection` methods: `getConnection()` and `getConnection(String, String)`.

148. The `getConnection()` method returns a connection created using the String, username, and password used to create the DataSource.

149. The `getConnection(String, String)` method returns a connection created using the DataSource (database), but with the username and password provided as arguments by the caller.

150. None of these methods for establishing a connection to a RAC instance allows the passing of a dynamic Web page request. Accordingly, JDBC never examines the request to make an informed selection of which RAC instance with which to establish the connection. JDBC simply executes the appropriate getConnection method and a RAC instance is selected irrespective of the URL of the Web page request.

151. Similarly, for connections made to a RAC instance using OCI and runtime load balancing, for example, OCI is going to select a session from a session pool for connecting to a particular RAC instance (as determined by the RAC Load Balancing Advisory) using different interfaces. There are three interfaces that can be used to log on to the database in connection/session pooling mode. Each one is described below:

152. `OCILogon2()` is used when a user needs a simple connection and does not need to alter any attributes of the session handle. This interface can also be used to make proxy connections to the database. Here is an example using `OCILogon2()`:

```
for (i = 0; i < MAXTHREADS; ++i)
{
      OCILogon2(envhp, errhp, &svchp[i], "scott",
5, "tiger", 5, poolName,
      poolNameLen, OCI_LOGON2_CPOOL));
```

```
}
```

153. `OCISessionGet()` is the Oracle recommended interface. It gives the user the additional option of using external authentication methods, such as certificates, distinguished name, and so on. Here is an example using `OCISessionGet()`:

```
for (i = 0; i < MAXTHREADS; ++i)
{
      OCISessionGet(envhp, errhp, &svchp, authp,
      (OraText *) poolName, strlen(poolName),
      NULL, 0, NULL, NULL, NULL,
      OCI_SESSGET_CPOOL)
 }
```

154. `OCIServerAttach()` and `OCISessionBegin()`: This interface can be used if the application needs to set any special attributes on the user session handle and server handle. For such a requirement, applications need to do this:

- Allocate all the handles (connection pool handle, server handles, session handles and service context handles).
- Create the connection pool.
- Call `OCIServerAttach()` with mode set to `OCI_CPOOL`.
- Call `OCISessionBegin()` with mode set to `OCI_DEFAULT` or `OCI_MIGRATE`.

155. None of these methods for establishing a connection to a RAC instance allow the passing a dynamic Web page request. Accordingly, OCI never examines the request to make an informed selection of which RAC instance with which to establish the connection. OCI simply executes the appropriate connection method and a RAC instance is selected without access to and irrespective of the content of the Web page request.

156. Oracle Database does not perform "dispatching" under Oracle's proposed claim construction for the same reasons discussed above. Just as JDBC and OCI do not "examine" a request, JDBC and OCI do not "analyze" a request to make an informed selection of which page

server should process the request. Neither JDBC nor OCI can "examine" or "analyze" a request, and they do not inform their selection of a RAC instance based on any information they have "examined" or "analyzed" as required by the parties' respective constructions. Therefore, Oracle Database does not perform "dispatching" under either party's proposed construction.

4. Database Does Not Have a Page Server that "Releases" a Web Server

157. Infringement of all the Asserted Claims requires the use of a "Web server" and a "page server" that "releases said Web server to process other requests." Dr. Finkel opines in his report that the accused "page server" of Oracle's Database product is a RAC instance. *See, e.g.,* A27, Exh. G, at pp. 2-3. Dr. Finkel also opines in his report that the accused Web server that is allegedly used with Oracle's Database is OHS (or the computer running OHS). Dr. Finkel provides no other examples of Web servers that can be used with Oracle Database to allegedly infringe the Asserted Claims.

158. It is my opinion that Oracle's Accused Database Products cannot be used to infringe any Asserted Claim because an Oracle RAC instance never "releases" OHS (or the computer running OHS) to process other requests as required by all of the Asserted Claims. As in my discussion of Internet Application Server above in ¶¶102-125, just as an OC4J application server never acts to release OHS, a RAC instance also never acts to release OHS. RAC instances never act to "release" or "free" OHS to do anything. Additionally, for the same reasons stated above, RAC does not act to "release" or "free" OHS via Reliable Network Protocol messages.

159. Additionally, under typical operating conditions, RAC instances are not communicating directly with OHS via a Reliable Network Protocol. Under such circumstances, a RAC instance could not send a message (e.g., a TCP message) to OHS that, according to Dr. Finkel, could "release" or "free" OHS to process other requests. When a RAC instance receives

a request from an OC4J server, for example, that instance would not send a network protocol message back to OHS. For the aforementioned reasons, it is my opinion that Database does not infringe the asserted claims because it does not perform the required "releasing" limitation.

160. The source code conclusively defines the structures and functions of the software programs that comprise the Oracle accused products. Based upon my analysis of the source code and technical documentation, and my conversations with Oracle engineers, as set forth in this declaration, it is my opinion that the accused products do not infringe epicRealm's asserted claims.

I declare under penalty of perjury under the laws of the United States that the foregoing is true and correct and that this declaration was executed on this 20th day of August, 2008 at One Shell Center, Houston, Texas.



Dr. Paul C. Clark

61468292 v1

08/21/2008 09:59 602--956-2111 FEDEX KINKO'S 0908 PAGE 01

CERTIFICATE OF SERVICE

I hereby certify that on August 28, 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 28, 2008 upon the following individuals in the manner indicated:

BY E-MAIL

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, Esquire
JENNER & BLOCK
330 N. Wabash Avenue
Chicago, IL 60611-7603

gbosy@jenner.com

/s/ James W. Parrett, Jr.

James W. Parrett, Jr. (#4292)

EXHIBIT A

PAUL C. CLARK
4705 Broad Brook Drive, Bethesda, Maryland, 20814
703.628.9500
paul@securemethods.com

SKILLS SUMMARY

A senior executive with extensive IT design, development and deployment experience, Dr. Clark has twenty years of direct technical and management experience in the computer and systems engineering environment. He is the Founder, President and CTO of SecureMethods Inc. His specialties include complex commercial development and deployment of scalable secure network and data processing systems. He has served as keynote speaker, expert witness for high profile commercial clients and before Congress. He has served on federal advisory committees and is an adjunct professor at The George Washington University.

EMPLOYMENT

President and CTO July 1999 - Present
SecureMethods Inc.
Bethesda, MD

- Serves as Managing Director
- Manages operations and sales staff.
- Manages commercial product development staff.
- Provides product design, development and deployment guidance.
- Provides sales engineering support and collected customer feedback.
- Defines and communicates strategic technical vision.
- Successfully directed the development and deployment of commercial products on multiple Windows and Unix Platforms

Chief Scientist Jan. 1995 – July 1999
DynCorp Network Solutions
Fairfax, VA

- Managed technical staff and deliverables for multiple large projects.
- Provided project design, development and troubleshooting guidance.
- Provided customer technical interface and problem resolution.
- Designed and deployed next generation architecture for high volume network database and storage systems.
- Created a suite of secure products marketed and sold to DoD and the Federal Government.
- Provided corporate-wide technical consultation and support.

Senior Security Engineer Sept. 1990 – Jan. 1995
Trusted Information Systems
Glenwood, Maryland

- Participated in the design and implementation of the reference implementation of Privacy Enhanced Mail (PEM) with public and secret key encryption to provide security services for electronic transmissions.
- Designed NIST's Smartcard API (SCAPI).. Implemented the SCAPI for the NIST 250 and utilized it to perform cryptographic operations for PEM.
- Implemented X.500 Certification and Distinguished Naming support for PEM. Functions supported included: CA and user registration, revocation, and high speed database for certificate storage and retrieval.
- Systems design and programming, including the TCB, for the Trusted Xenix and Trusted Mach MLS operating systems. Tasks included MAC labeling and audit strategies, as well as application development.
- Designed and implemented multilevel electronic mail and HTTP proxy for Trusted Mach. Task included cryptographic support for the TCB as well as application specific validation and semantic checking for up/downgrade.
- Inventor of the Boot Integrity Token System (BITS), which provides hardware, enforced authentication within the boot sequence and guarantees operating system integrity.

Technical Lead Nov 1989 – Sept 1990
GTE Government Systems
Rockville, MD

- Managed the SAFE91 testbed effort with responsibilities including: budgets, schedules, customer negotiations, briefings, training, and tasking of technical staff. The testbed was created to simulate the client environment for the purpose of evaluating software packages and platforms for use by the client.
- Developed network load simulations for OS/2 LAN Manager to determine performance characteristics during periods of heavy traffic. This task included the design and development of a multi-threaded connection server under OS/2 version 1.0
- Designed and implemented an X Windows interface for the Minstrel System. This included the individual development of 20,000 lines of code in less than two months.
- Developed and taught DEC Windows and X Windows classes for GTE technical personnel. Responsible for instruction and problem solving throughout the subsequent development cycle.

Systems Engineer May 1985 – Nov. 1989
Ultrasystems Defense and Space

- Redesigned the Morse Mission Trainer while coordinating and tasking a team of programmers. During this time, received the President's quarterly award for outstanding performance. Task included systems and network and kernel level programming on SCO Xenix.
- Designed and implemented a database file server, benchmarked at over 100 retrievals per second on a million entry database. System implemented B* trees in C and ran on a Sun 3/260 workstation.
- Designed and implemented a satellite mission scheduler for multiple vehicles with multiple resources. Task included defining areas of interest (AOIs), flight paths, and optimal time allocation of resources.
- Designed and implemented the Ultraplot graphics tool to produce line and scatter plots, as well as bar graphs and histograms from large datafiles. This utility was implemented utilizing the DI3000 (device independent) graphics package.

HARDWARE

Mainframe, Workstation, PC, including: IBM, DEC, Sun, HP, SGI, Intel

SOFTWARE

UNIX (Linux, System V, BSD, Solaris, AIX, IRIX, Xenix), MS DOS/Windows/95/98/NT/2000/XP, VMS, OS/2, X Windows, DEC Windows, Presentation Manger, TCP/IP, X25, Xenix Net, IBM LAN, DEC Net, Ethernet, Token Ring

EDUCATION

DSc. in Computer Science
Concentration in Security, Graphics, Intellectual Property Law
The George Washington University, 1994

M.S. in Electrical Engineering and Computer Science
University of Southern California, 1988

B.S. in Mathematics
University of California Irvine, 1986

Graduate level study in all major areas of computer science

REPRESENTATIVE PUBLICATIONS

"BITS – A Smartcard Protected Operating System," with Lance Hoffman, Communications of the ACM, November 1994.

"Service Layering Promotes Secure Data Exchange in Diverse Environments," Computer News, October 23, 1995

"Threats Posed to Cryptographic Applications by Random Numbers," presented to the RSA Data Security Conference, January 1996.

"A Reference Model for Electronic Commerce," with Daniel J. Blum and John Jauregui, Messaging Magazine, December 1996, Volume 2, Number 7.

"Secure Compartmented Data Access over an Untrusted Network Using a COTS-based Architecture," with Marion C. Meissner, and Karen O. Vance, Presented to the Annual Computer Security Applications Conference (ACSAC'00), New Orleans, December, 2000. Later published in "Statistical Methods in Computer Security," Marcel Dekker, ISBN 0-8247-5939-7, edited by William W. S. Chen, 2005

ADDITIONAL INFORMATION

(1) Dr. Clark was a member of the Federal Advisory Committee for Key Management Infrastructure (KMI); he was Chairman of the Interoperability Working Group for Cryptographic Key Recovery.

(2) Dr. Clark served as a Cooperative Research and Development Agreements (CRADA) partner, which is a joint effort between the National Institute of Standards and Technology (NIST) and several companies formed to begin development of the elements of a Public Key Infrastructure (PKI). A core element of this effort is the development of a Minimum Interoperability Specification for PKI components MISPC.

(3) Dr. Clark serves as an adjunct professor in the Electrical Engineering and Computer Science Department at The George Washington University. He teaches doctoral level cryptography and computer security courses.

(4) Speaker at The Federal Information Security Conference; presented "Embedded Security Deployment," Colorado Springs, CO, March 31, 2006

(5) Keynote Speaker for the Washington DC Bar Association; presented "Security for the Networked Computing Environment," August 8, 2005

(6) Appeared before Congressional committee to provide expert testimony; presented "Advanced Technology for Border Control," July 23, 1998.

(7) Keynote speaker at Mass Storage Conference; presented "Secure Data Access Over Public Networks," New Orleans, LA, October, 1996

(8) Keynote speaker at health care convention, presented "Security for Health Care Records," Nashville, TN, May, 1997.

(9) Keynote speaker at the USDA Plant and Genome Conference; presented "A Secure Architecture for Data and Systems," Nimes, France, October, 1997

(10) Speaker at IEEE Technical Meeting; presented "A Comprehensive Security Architecture," Virginia, June, 1996.

RECENT CASES

Lead expert for Lucent - Microsoft v Lucent - Pending

Lead expert for Oracle - Mangosoft v Oracle – Summary judgment in favor of Oracle

Lead expert for RSA Data Security – Digital Privacy v RSA – Summary judgment in favor of RSA at the Markman hearing

EXHIBIT B

Materials Considered

1. U.S. Patent No. 5,894,554 Patent-in-suit, and its prosecution file history
2. U.S. Patent No. 6,415,335 Patent-in-suit and its prosecution file history
3. U.S. Patent No. 5,754,772 (Leaf)
4. 10.1.3.1 Application Server Control Metric Computations, 03/01/2005, ORCL 01251768-01251783
5. Chapter 1: The Dispatcher (valid for label RDBMS_8.1.3_S0LARlS_980815), undated, ORCL 01268094-01268112
6. Design Specification Design Specification for Metric Collection Engine, OEM, 10gR2, 03/04/2004, ORCL 01285982-01285995
7. Design Specification for Enhanced Listener, SQL*Net, 3.0, 04/08/1996, ORCL 01274261-01274281
8. Design Specification for Listener Load Balancing, Net8, 8.1, 12/01/2007, ORCL 01329652-01329662
9. Design Specification for OCI Runtime Connection Load Balancing in 11g, 12/14/2005, ORCL 01310468-01310498
10. Design Specification for ODP.NET RAC and Grid Optimizations, 10gR2, 05/11/2004, ORCL 01285488-01285503
11. Design Specification for Server - Side Runtime Load Balancing, RDBMS, 10.2, 04/29/2007, ORCL 01289835-01289845
12. Functional Specification Configuration Specification for RAC High Availability, Oracle9i Release 2, 07/15/2001, ORCL 01150370-01150452
13. Functional Specification for Apache Configuration and Management, Ver. 0.8, 06/27/2000, ORCL 01146238-01146301
14. Functional Specification for Enhanced Listener, SQL*Net, 3.0, 04/08/1996, ORCL 01098385-01098397
15. Functional Specification for Enhancement of Load Balancing in mod_oc4j, 06/05/2002, ORCL 01122202-01122213
16. Functional Specification for Inclusion of DMS statistics in mod_oc4j, Oracle HTTP Server, 9.0.4, 06/06/2002, ORCL 01122193-01122201
17. Functional Specification for Listener Load Balancing, Net8, 8.1, 10/06/1997, ORCL 01110194-01110206
18. Functional Specification for ODP.NET RAC and Grid Optimizations, 10gR2, 05/06/2004, ORCL 01052550-01052564

19. Functional Specification for PL/SQL Server Pages, PL/SQL, 8.1.6, Ver. 12/20/99, 12/20/1999, ORCL 01113690-01113722

20. Functional Specification for RAC High Availability, Oracle 9i Release 2, 04/25/2001, ORCL 01150453-01150556

21. Functional Specification for Runtime Load Balancing in OCI Session Pools, 11gRl, 12/02/2005, ORCL 01166752-01166776

22. Functional Specification for Runtime Load Balancing, RDBMS, 10gR2, Ver. 6.0 04/01/04, 04/01/2004, ORCL 01060148-01060193

23. Functional/Design Specification for Runtime Work Request Load Balancing using RAC Service Metrics, JDBC, 10gR2, Ver. 1.1, 05/01/2004, ORCL 01054844-01054858

24. Managing Content with Oracle XML DB, An Oracle White Paper, 03/00/2005, ORCL 00933137-00933174

25. Multiple JVM support in 10.1.3 SOA Metric-related Changes, 01/31/2006, ORCL 01251784-01251791

26. Oracle 9iAS Best Practices in PSP Development, 08/27/2001

27. Oracle Application Developer's Guide - XML 10g (9.0.4) BI2099-01, 09/00/2003, ORCL 00380115-00381470

28. Oracle Application Server 10g Best Practices 10g (9.0.4) B12223-01, 05/00/2004, ORCL 00384158-00384305

29. Oracle Application Server 10g Concepts 10g (9.0.4) B10375-02, 03/00/2004, ORCL 00370687-00370914

30. Oracle Application Server 10g High Availability Guide 10g (9.0.4) B10495-02, 03/00/2004, ORCL 00375949-00376112

31. Oracle Application Server Administrator's Guide 10g Release 2 (10.1.2) B13995-04, 8/00/2005, ORCL 00440876-00441519

32. Oracle Application Server Administrator's Guide 10g Release 3 (10.1.3.2.0) B32196-01, 1/00/2007, ORCL 00725652-00726027

33. Oracle Application Server Best Practices Guide 10g Release 2 (10.1.2) B28654-01, 03/01/2006, ORCL 00528055-00528218

34. Oracle Application Server Concepts 10g Release 2 (10.1.2) B13994-02, 7/00/2005, ORCL 00440680-00440875

35. Oracle Application Server Containers for J2EE Servlet Developer's Guide 10g (9.0.4) B10321-01, 09/00/2003, ORCL 00338102-00338415

36. Oracle Application Server Containers for J2EE, Support for JavaServer Pages Developer's Guide, 10g (9.0.4) B10320-01, 09/00/2003, ORCL 00337734-00338101

37. Oracle Application Server Containers for J2EE, Support for JavaServer Pages Developer's Guide, 10g Release 2 (10.1.2) B14014-02, 07/00/2005, ORCL 00444343-00444558

38. Oracle Application Server Globalization Support Guide 10g Release 2 (10.1.2) B14004-02, 07/00/2005, ORCL 00442863-00443012

39. Oracle Application Server mod_plsql User's Guide 10g (10.1.3.1.0) B28963-01, 09/00/2006, ORCL 00664145-00664224

40. Oracle Application Server Web Cache Administrator's Guide 10g Release 2 (10.1.2) B14406-01, 12/00/2004, ORCL 00473184-00473739

41. Oracle Call Interface Programmer's Guide 11g Release 1 (11.1) B28395-02, 10/00/2007

42. Oracle Containers for J2EE Configuration and Administration Guide 10g (10.1.3.1.0) B28950-01, 10/00/2006, ORCL 00661493-00661764

43. Oracle Database Administrator's Guide 10g Release 2 (10.2) B14231-02, 05/00/2005

44. Oracle Database 10g Release 2 XML DB, An Oracle Technical White Paper, 5/00/2005, ORCL 00933009-00933105

45. Oracle Database 2 Day + Real Application clusters Guide 10g Release 2 (10.2) B28759-01, 12/00/2006

46. Oracle Database High Availability Overview 11 g Release 1 (11.1) B28281-0 1, 7/00/2007

47. Oracle Database JDBC Developer's Guide and Reference 10g Release 2 (10.2) B14355-02, 1/00/2007

48. Oracle Database JDBC Developer's Guide and Reference 10g Release 2 (10.2) B14355-02, Ch. 24 Run-Time Connection Load Balancing, 06/29/1905

49. Oracle Database JDBC Developer's Guide and Reference, 11g Release 1 (11.1) B31224-03, 9/00/2007

50. Oracle Database Net Services Administrator's Guide 10g Release 2 (10.2) B14212-02, 10/00/2005

51. Oracle Database Net Services Administrator's Guide 10g Release 2 (10.2) B14212-02, Ch. 3 Connectivity Concepts, 06/27/2005

52. Oracle HTTP Server Administrator's Guide 10g Release 2 (10.1.2) B14007-03, 7/00/2005, ORCL 00443013-00443294

53. Oracle Net Services for Oracle 10g, An Oracle White Paper, 05/00/2005

54. Oracle Real Application Clusters 11g, An Oracle Technical White Paper, 04/00/2007

55. Oracle Real Application Clusters Administration and Deployment Guide 11g Release 1 (11.1) B28254-04, 11/00/2007

56. Oracle Web Cache Administration and Deployment Guide Release 1.0.2 A86722-01, 11/21/2000, ORCL 00107955-00108126

57. Oracle XML DB Developer's Guide 11g Release 1 (11.1) B28369-02, 10/00/2007

58. Oracle7 Server Concepts Release 7.3 A32534-1, 02/00/1996

59. Oracle8i JDBC Developer's Guide and Reference Release 3 (8.1.7) A83724-01, 07/00/2000,

ORCL 00100977-00101588

60. Oracle9i Application Server Best Practices Release 1 (v1.0.2.2) A95201-01,9/00/2001, ORCL 00190234-00190397

61. Oracle9i Application Server Component Diagram, undated, ORCL 00110683

62. Oracle9i Application Server Overview Guide Release 1.0.2 A87353-0 1, 11/00/2000, ORCL 00108517-00108636

63. Oracle9i Application Server Performance Guide Release 2 (9.0.2) A95102-02, 4/00/2002, ORCL 00847081-00847294

64. Oracle9i Application Server Release 2 New Feature Summary, An Oracle Technical White Paper, 9/00/2002, ORCL 00010874-00010914

65. Oracle9i Application Server: Availability , Scalability, and Manageability of J2EE and Web Cache Clusters, An Oracle White Paper, 5/00/2002, ORCL 00013359-00013398

66. Oracle9i Application Server: mod_oc4j Technical Overview, 5/00/2002

67. Oracle9i JDBC Developer's Guide and Reference Release 2 (9.2) A96654-01, 03/00/2002, ORCL 00306997-00307580

68. Oracle9i Net Services Administrator's Guide Release 2 (9.2) A96580-02, 10/00/2002, ORCL 00875561-00876162

69. Oracle9iAS Web Cache Administration and Deployment Guide Release 2 (9.0.2) A95404-02, 05/00/2002, ORCL 00272342-00272887

70. Oracle9iAS Web Cache: Administration and Deployment Guide Release 2.0.0 A90372-04, 10/10/2001, ORCL 00189864-00190165

71. Presentation, 8 Simple Caching Rules: Web Caching Your Applications for Fast Response Time, 12/09/2004, ORCL 00021413-00021455

72. Presentation, Oracle9iAS Web Release: New Features in Release 2 (9.0.4), undated, ORCL 00026552-00026590

73. Put Your Cache Where It Belongs, undated, ORCL 00848006-00848043

74. ORCL -SC 000001-001396

75. Tannenbaum, Computer Networks, Prentice-Hall (1981)

76. Stevens, TCP/IP Illustrated, Vol. 1, Addison-Wesley (1994)

77. Williams & Stevens, TCP/IP Illustrated, Vol. 2, Addison-Wesley (1995)

78. Stevens, UNIX Network Programming, Vol. 1 (2nd ed.), Prentice-Hall (1998)

79. Deposition Transcript of Nipun Agarwal, 12/20/2007, and exhibit Agarwal 1

80. Deposition Transcript of Michael Bogue, 05/16/2008, and exhibits Bogue 1-3

81. Deposition Transcript of Joseph Chandy, 12/21/2007

82. Deposition Transcript of Sum ant a Chatterjee, 12/20/2007, and exhibits Chatterjee 1-4

83. Deposition Transcript of Lakshminarayanan Chidambaran, 02/29/2008, and exhibits Chidambaran 1-10

84. Deposition Transcript of Mark Clark, 04/17/2008, and exhibits Clark 1-6

85. Deposition Transcript of Carol Colrain, 04/15/2008, and exhibits Colrain 1-8

86. Deposition Transcript of Clark Elms, 05/13/2008, and exhibits Elms 1-8

87. Deposition Transcript of Harvey Eneman, 05/14/2008, and exhibits Eneman 1-10

88. Deposition Transcript of Steve Harris, 04/18/08, and exhibits Harris 1-15

89. Deposition Transcript of Hal Hildebrand, 12/21/2007, and exhibit Hildebrand 1

90. Deposition Transcript of Craig Holmes, 03/12/2008, and exhibits Holmes 15

91. Deposition Transcript of Pushkar Kapasi, 02/27/08, and exhibits Kapasi 1-4

92. Deposition Transcript of Mohammad Lari, 02/27/2008, and exhibits Lari 1-5

93. Deposition Transcript of Andrew Levine, 04/17/2008, and exhibits Levine 1-35

94. Deposition Transcript of Andrew Levine, 05/01/2008, and exhibits Levine 36-48

95. Deposition Transcript of Keith Lowery, 04/25/2008, and exhibits Lowery 1-16

96. Deposition Transcript of Keith Lowery, 04/26/2008, and exhibits Lowery 17-56

97. Deposition Transcript of Jun Makino, 05/16/2008, and exhibits Makino 1-7

98. Deposition Transcript of Stephen Mayer, 02/26/2008, and exhibits Mayer 1-3

99. Deposition Transcript of Matthew Mayerson, 04/15/2008

100. Deposition Transcript of Katrina Montinola, 02/19/2008, and exhibits Montinola 1-13

101. Deposition Transcript of Mark Nelson, 02/22/2008, and exhibits Nelson 1-18

102. Deposition Transcript of Kant Patel, 02/25/2008, and exhibits Patel 1-12

103. Deposition Transcript of Peter Povinec, 04/16/2008, and exhibits Povinec 1-6

104. Deposition Transcript of Matthew Sarboraria, 05/15/2008, and exhibits Sarboraria 1-3

105. Deposition Transcript of Douglas Surber, 02/28/2008, and exhibits Surber 1-9

106. Deposition Transcript of Timothy Willis, 05/21/2008, and exhibits Willis 1-28

107. All interrogatories propounded by both parties and all responses to date

108. epicRealm Proposed Claim Constructions

109. Oracle Proposed Claim Constructions

110. All filings related to epicRealm's Claim Construction Brief dated July 31, 2008

111. All filings related to epicRealm's Motion for Partial Summary Judgment that Oracle

References Do Not Anticipate dated July 31, 2006

112. All filings related to epicRealm's Motion to Exclude Testimony of Dr. Paul Clark dated July 31, 2008

113. All filings related to epicRealm's Motion to Exclude Testimony of Terry Musika dated July 31, 2008

114. All filings related to epicRealm's Motion to Exclude Testimony of Dr. Michael Shamos dated July 31, 2008

115. All filings related to epicRealm's Motion for Summary Judgment of Literal Infringement dated July 31, 2008

116. Declaration of Dr. David Finkel in Support of Defendant's Claim Construction Brief, Motion for Partial Summary Judgment of Literal Infringement and Motion for Partial Summary Judgment that the Oracle References Do Not Anticpated dated July 31, 2008

61385458 v1